AN ABSTRACT OF THE THESIS OF

Trent L. McDonald for the degree of Doctor of Philosophy in Statistics presented on May 6, 1996. Title: Analysis of Finite Population Surveys: Sample Size and Testing Considerations.

Abstract approved

N. Scott Urquhart

This dissertation concerns two topics in the analysis of finite population surveys: setting sample size and hypothesis testing. The first concerns the *a priori* determination of the sample size needed to obtain species members. The second concerns testing distributional hypotheses when two equal-size populations are sampled.

Setting sample size to obtain species is a problem which arises when an investigator wants to obtain (1) a member of all species present in an area (2) a member of all species whose relative frequency is greater than, say, 20% or (3) a member of each species in a target set of species. Chapter 2 presents a practical solution to these questions by setting a target sample size for which the species are obtained with known probability. The solution requires the estimated relative frequency of the rarest species of interest; total number of species is not needed. Because this problem has substantial computational demands, easy-to-compute formulas are needed and given. Three practical examples are presented.

Testing of finite population distributional hypotheses is covered in Chapter 3. The test proposed here works under reasonably general designs and is based on a Horvitz-Thompson type correction of the usual Mann-Whitney U statistic. The investigation here compared this proposed test to a corrected (for finiteness) form of the usual Wilcoxon rank sum test. Size and power of the two test procedures are investigated using simulation. The proposed test had approximately correct nominal size over a wide range of situations. The corrected Wilcoxon test exhibited extreme violations in size in many cases. Power of the two tests in situations where they have equal size is similar in most practically interesting cases.

Analysis of Finite Population Surveys: Sample Size and Testing Considerations

by

Trent L. McDonald

A THESIS

submitted to

Oregon State University

in partial fulfillment of

the requirements for the

degree of

Doctor of Philosophy

Presented May 6, 1996

Commencement June 1996

Doctor of Philosophy thesis of Trent L. McDonald presented on May 6, 1996.

APPROVED:

Redacted for Privacy

Major Professor, representing Statistics

Redacted for Privacy

Chair of Department of Statistics

Redacted for Privacy

Dean of Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Trent L. McDonald, Author

ACKNOWLEDGMENT

I wish to thank my major professor, Dr. Urquhart, for all his help, guidance, and friendship. I highly recommend him to anyone contemplating a Ph.D. in statistics. I also wish to thank my committee, especially Dr. Birkes, for their time and continued interest. All funding for this research was procured from the U.S. Environmental Protection Agency and I wish to thank two people responsible for that, Dr. Scott Urquhart and Dr Tony Olsen. My children, Zoe and Aidan, were wonderful distractions throughout this process and a prime source of motivation. Finally, I owe a big debt of gratitude to my wife, Mary. This process was not easy for her but in the end she was there for me and the kids.

CONTRIBUTION OF AUTHORS

Dr. Scott Urquhart proposed the original questions which eventually lead to both manuscripts. Dr. Urquhart was also involved in proposing solutions, interpreting results, and editing of each manuscript. Dr. David Birkes was involved in the second manuscript. He contributed to its solution and was involved in editing the manuscript.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF TABLES

To my father.

Analysis of Finite Population Surveys: Sample Size and Testing Considerations

Chapter 1

Introduction

This thesis is concerned with two practical aspects of finite population surveys, (1) the acquisition of species members, and (2) the testing of a distributional hypothesis when sampling from two equal-size populations. The original motivation to work on these topics was a need to solve a pair of applied problems for which little information could be found. As a result, it is hoped that the solutions presented here are practical in nature and simple enough that they might be applied in a wide range of studies. Each topic is worthwhile studying by itself and each is presented in stand-alone chapters so that they might be applied separately. Often, however, the topics covered here will be applicable in the same study. It is conceivable that researchers would be interested in setting a desirable sample size, and having done that, test for differences between two sampled populations.

The first topic, obtaining members of a certain species, was motivated by the desire to somehow quantify the number and type of species present in a set of sampled material or population (so called 'species richness'). An absolute answer to this question could, and can, be obtained in many situations where it is possible to simply enumerate all individuals in a population and record their species. A practical problem arises in enumeration when the population is too large or too difficult to deal with physically. However, unless the entire population is enumerated (censused), there is no way to be certain all species have been counted. Certainly common species will be found, but how many rare species will be found if, say, half the population is examined?

Chapter 2 proposes a simple, practical solution to the problem of obtaining species. To arrive at the solution, the problem is first viewed as one of computing a probability

level for the situation. That is, the goal is to set individual sample size so that a certain number of species are obtained with known probability. The solution to this probability statement is itself plagued by practical problems. One practical problem plaguing the solution is lack of adequate computing resources. The problem can be posed and a mathematical solution found, but it cannot be computed on current computers in our lifetimes. An approximation is needed which can be computed in a relatively short time and still provide useful information. Chapter 2 discusses the exact sample size and proposes two approximations to it. As a first step in both approximations, a probability model (density) from an infinite population is used to approximate a probability model defined on a finite population. The hope is that populations are large enough that the approximation is adequate. So, while the focus of this thesis is on *finite* populations, which are real, solutions derived herein draw upon methods from classical (infinite) population statistical theory.

Chapter 3 of this thesis deals with the practical problem of hypothesis testing under general designs. Hypothesis testing under general survey designs is a broad field encompassing nearly all topics covered by infinite population hypothesis testing plus others not covered in the infinite case. Issues in testing include test size, test power, non-parametric tests, parametric tests, appropriate distributions under the null hypothesis, etc. This broad field is narrowed somewhat by proposing a non-parametric test which tests equality of distribution functions and which performs adequately under a wide range of survey designs. By no means is this test the only test which could be defined, nor is it required to perform well under all sample designs and populations. The approach taken in Chapter 3 is to define a test and show that it performs better than naively applying an infinite population test. Test size is the primary concern in evaluating the performance of the tests. Power to detect two types of alternatives is also investigated.

The test proposed in Chapter 3 is an easy-to-compute, practical test based on ranks which tests for differences in underlying cumulative distribution functions. The test can be thought of as the finite population cousin of the (infinite population) two-sample Wilcoxon rank sum test (Wilcoxon, 1945). Focus is restricted to cases where

the two populations being compared are the same size. These cases most commonly arise when the same set of material is sampled at two points in time, but also arise in other situations. For example, two separate computer chip wafers might be sampled by selection of a sample of chips from each. Because each wafer yields the same number of chips, the sampled populations are the same size. As in Chapter 2, applications of the test proposed in Chapter 3 is complicated by practical considerations. The primary difficulty in Chapter 3 is the difficulty involved in evaluating the test's performance under all possible survey designs. The test's performance is evaluated by computer simulation and as a result, test properties under situations outside those simulated are unknown.

Each chapter of this thesis, except this and Chapter 4 is a self-contained unit, complete with its own introduction, notation, conclusions, etc. As such, parts of this thesis can be read in pieces and in any order. Chapter 4 is a summary of the work and directions for future research.

Chapter 2

Obtaining Species: Sample Size Considerations

Trent L. McDonald, David S. Birkes, and N. Scott Urquhart

Submitted to *Environmental and Ecological Statistics*

Center for Statistical Ecology and Environmental Statistics, Penn State University,
421 Classroom Building, University Park, PA 16802

October 1995, 24 pages.

Abstract

Suppose fish are to be sampled from a stream. A fisheries biologist might ask one of the following three questions: "How many fish do I need to catch in order to see all of the species?", "How many fish do I need to catch in order to see all species whose relative frequency is more than 5%?", or "How many fish do I need to catch in order to see a member from each of the species A, B, and C?". This paper offers a practical solution to such questions by setting a target sample size designed to achieve desired results with known probability. We present three sample size methods, one we call "exact" and the others approximate. Each method is derived under assumed multinomial sampling, and requires (at least approximate) independence of draws and (usually) a large population. The minimum information needed to compute one of the approximate methods is the estimated relative frequency of the rarest species of interest. Total number of species is not needed. Choice of a sample size method depends largely on available computer resources. One approximation (called the "Monte Carlo approximation") gets within ± 6 units of exact sample size, but usually requires 20-30 minutes of computer time to compute. The second approximation (called the "ratio approximation") can be computed manually and has relative error under 5% when the all species are desired, but can be as much as 50% or more too high when exact sample size is small. Statistically, this problem is an application of the "sequential occupancy problem." Three examples are given which illustrate the calculations so that a reader not interested in technical details can apply our results.

Keywords: multinomial distribution, occupancy problem, species richness, urn model

2.1 Introduction

A fisheries biologist inquires, "How many fish do I need to catch in order to see all the species?" Upon reflection, it becomes apparent that no practical answer to this question guarantees all species will be seen. It usually is neither practical nor permissible to collect all the fish in a stream. Even if an entire reach of stream were poisoned and fish carcasses collected at the bottom, a rare species could die but be trapped by its habitat and never caught. Due to inherent randomness in the problem, there is no way to be absolutely certain that all species will be seen if a sample of fish is drawn. Further, the sample size required to have appreciable probability of obtaining a rare species can be impossibly large.

What can the biologist do? One approach is to define both what is "rare" and the desired chance of success in meeting a desired outcome, then attempt to determine sample size such that all species at least as common as the "rare" ones are captured with the desired confidence. Practically, this amounts to allowing species rarer than a specific rarity to be ignored. This leads us to investigate questions like, "If the rarest species of interest occurs at a rate of 5 per 1000, how many fish will have to be caught in order to have 90% confidence that all species of interest will be seen?" We offer an answer to this question provided the biologist does not catch a significant fraction of the population and the fish are (at least approximately) randomly distributed in their habitat.

Here, we present three distinct ways to evaluate the sample size required to obtain, with known confidence, at least one member of some number of species or at least one member of each of some set of species assuming that a multinomial sample is drawn from a population. We term the three sample sizes "the exact sample size," "the Monte Carlo approximation," and "the ratio approximation." Choice of which sample size method to use depends on available time and computer resources. If a sample size is needed immediately and/or no computer is available, the ratio approximation should be used; it provides a good approximation in many cases. If more time and a computer is available, the Monte Carlo approximation should be used as it provides a

better approximation than the ratio approximation, generally within ± 6 units of the exact number. If substantially more time exists and computer resources are available, the exact method should be used because the other methods only approximate this number.

Strictly speaking, our assumption of multinomial sampling requires: An infinite supply of individual organisms is available in the population; and that the species membership of any individuals in the sample be statistically independent. Assuming that organisms are randomly distributed in their habitat allows the multinomial distribution to be used when the organisms are captured in batches, as in a seine haul or area (quadrat) sample. If organisms are randomly picked one-at-a-time from a population, the assumption of a randomly distributed (in time or space) population is not needed. In some situations, the independence assumption can be met by thoroughly mixing the population of interest. The multinomial and assumptions are considered further in Section 2.5.

Questions concerning sample size required to obtain at least one member of each of some set of classes also arise outside the biological sciences. Practically every field has had to grapple with setting sample size to obtain membership. Although a common problem, we will emphasize its application in ecology and the biological sciences by focusing on obtaining one or more members of some set of species. Among others, we relate an example of an aquatic study of *macrobenthos* (bottom dwelling insects) where, given the time and effort necessary to identify these small creatures, it is desirable to reduce total effort by identifying only enough specimens to be reasonably confident at least one member of some subset of species (or *taxa* in this case) is obtained.

Statistically, obtaining species can be modeled using an "urn model," and sometimes has been called the "sequential occupancy" problem. The occupancy problem has been around for a long time and the amount of literature devoted to it is large. Johnson and Kotz (1977) is an essential modern reference for urn models and the occupancy problem. Certain occupancy problems focus on different types of occupancy (different types of balls), calculating probability of k balls occupying y urns, and pro-

vide for certain types of non-independence in occupancy (i.e., in the balls). We are interested in occupancy of at least k=1 balls in some number of urns where the balls are independent. Much writing has been devoted to the "classical" occupancy problem wherein each class (species) has equal likelihood of occurring. This case provided an early model for occupancy, but serves only as a relatively uninteresting case in the present setting which requires that classes (species) be allowed to occur with varying probabilities. Nath (1974) presents a probability mass function (*pmf*) which gives the probability of obtaining some number of classes in a specific number of draws from a general population; a result directly relevant to this paper (see also Johnson and Kotz, (1977)). Nath also gives the expected number of samples and variance of the distribution of sample size. Although this *pmf* has been known for some time, it has not seen widespread application because it is rather complicated. Moreover, little guidance is given regarding its use and specifically how it might be used to calculate sample size. Even if someone knew how to apply the *pmf* to calculate sample size in a particular situation, the time (computer time) required is prohibitive even for relatively small problems. Consequently, approximate or asymptotic results which speed calculations are valuable. Johnson and Kotz (1977) and Kolchin et al. (1978) presented asymptotic distributions of sample size in situations were the ratio of sample size to population size converges to some number (e.g., zero or infinity). In this paper, we assume the population size is large and take a non-asymptotic approach to approximating sample size.

A related topic, which we will not pursue, is the estimation of the (unknown) total number of species. Bunge and Fitzpatrick (1993) give a through review of these techniques and conclude that no single best estimation technique exists.

The remainder of this article is organized as follows: Section 2.2 presents three examples, in increasing order of complexity. They are intended to illustrate sample size computations so a reader not interested in derivation details can apply our results. In Section 2.3 we describe calculation of exact sample size under the multinomial and develop our approximations. Section 2.4 presents investigations into the accuracy of the ratio approximation, and Section 2.5 contains a discussion of the multinomial and

assumptions. Derivation of the underlying probability function used throughout the paper is relegated to an appendix.

2.2 Examples

We intend for the following examples to illustrate calculations well enough that our result can be applied without encumbering details. The first example approximates sample size needed to obtain a member of each species in some target set of species, the second example approximates sample size required to see all species whose relative frequency is over a specified level, and the third example approximates sample size when full pilot data is available. The first two examples require a minimum amount of information and illustrate special cases of the ratio approximation derived in Section 2.3.3. Notation is consistent between this and Section 2.3.3.

2.2.1 Obtaining Specific Species

Suppose there exists a target set of species. Our goal is to obtain at least one member of each species in the target set with prescribed confidence, say γ. Assume that individual's species membership is identified in random order. The minimum information needed to approximate sample size required to meet our goal is (1) an estimate of the relative frequency of the rarest species in the target set of species (denote this number by π^*), and (2) the number of species having relative frequency "close to" the rarest both inside and outside the target group of species (denote this by r^*). If more than the minimal information is known, see the third example (Section 2.2.3). Given π^* and r^*, the approximate sample size required to obtain at least one member of all species in the target set with γ confidence is,

$$n \approx \frac{ln(1-\gamma^{1/r^*})}{ln(1-\pi^*)} \tag{2.1}$$

Consider the following example. Researchers have found two new and extremely rare grass species on study area A. The researchers now wish to know if either species is present on study area B. The researchers hypothesize that, if present, the two new species would be the least frequent species in the study area and that there might exist a third rare species of similar frequency in the study area. Assuming three rare species are present in study area B, the researchers wish to, with 95% confidence, obtain a specimen from all three if all three species occur with relative frequency greater than 1 in 1000. That is, if researchers fail to obtain a specimen of any of the three rare species, researchers wish to state with (approximately) 95% confidence that, if present, the unobtained rare species occurs on study area B at a rate less than 1 in 1000. Setting π^*=0.001, r^*=3, and γ=0.95 in Equation 2.1 yields a sample size of $n \approx ln(1 - 0.95^{1/3})/ln(1 - 0.001) = 4,075$ individual grass plants.

In the above example, researchers would need to insure that *individual grass plants* are obtained at random. If plants are obtained via quadrat or area sampling (e.g., by clipping all plants within a one meter frame), the researchers would be making an *implicit* assumption that individual grass plants enter the quadrats essentially at random. Effectively, they are assuming that the population of interest (grass plants) is not "clumped" or "patchy" and that individuals are distributed at random. To reduce the effects that "clumping" has on sample size in such area sampled populations, sampling units should be made as small as feasibly possible (ideally, one individual is captured in each sampling unit).

2.2.2 Large Volumes of Sample Material

Suppose researchers have available a sample of material which is so large that they cannot hope to process all of it. We have seen several real examples of this type, including: aquatic invertebrate samples, entomological samples, samples gathered with automatic data collection equipment, and massive data sets from supermarkets and telephone companies. A relatively common goal in such studies is documentation

of the presence of different "types" in the sample. In keeping with our emphasis on biological populations, we will continue to equate "types" with "species."

At times, it may be useful to set sample size (number of individuals) so that, with known probability, at least one individual is obtained from every species whose relative frequency is greater than or equal to π^*, where π^* is some number between 0 and 1. A conservative (too high) approximation to a sample size meeting this goal is obtained by setting $r^* = [1/\pi^*]$ in Equation 2.1, where "$[1/\pi^*]$" denotes the integer portion of $1/\pi^*$. The information specified in this case is π^* and γ.

Accuracy of the approximation can be improved with more knowledge. Recall from the first example that r^* is the number of species with relative frequency "close" to π^*. If general information about k, the number of species, is known, accuracy of the approximation can be improved by setting $r^*=min(k, [1/\pi^*])$ in Equation 2.1. Additionally, r^* can be estimated directly, or at least bounded above, from knowledge of the population.

Consider the following example. Researchers have available a trash-can sized container of sediment containing (presumably) thousands of insects. The researchers would like to identify at least those species which comprise more than 10% of the total sample. That is, researchers would like to identify enough insects so that they obtain, with 95% probability, a specimen from at least those species whose relative frequency is greater than 10%. Setting γ=0.95, π^*=0.10, and r^*=10 in Equation 2.1 indicates that approximately $n \approx ln(1-0.95^{0.1})/ln(1-0.1) = 50.07 \approx 50$ insects need to be individually identified in order to obtain all species whose relative frequency is greater than or equal to 10% with 95% confidence.

Complete mixing of the container of insects is required to ensure insects are identified in random order and so that the computed (approximate) sample size of 50 is accurate. If no mixing takes place, actual sample size to achieve the goal may be substantially higher than 50.

Continuing this example, suppose researchers have more knowledge about the population. Suppose researchers feel confident hypothesizing that two dominant species together account for at least 80% of the population. This means at most two other

species have relative frequencies close to 10%. With this knowledge and setting γ=95%, π^*=10%, and r^*=2 in Equation 2.1, researchers can revise their estimate of sample size to n=35 individually identified insects in order to obtain all species with relative frequency greater than 10%.

Note that if sample size is set as above, it is entirely possible to obtain a member of any species, regardless of its relative frequency. There also exists a subtle distinction in interpretation at the end of sampling: If species A, B, and C are obtained, then with (approximately) γ confidence, no other species in the population has relative frequency greater than or equal to π^*; it is not true, however, that if species A, B, and C are obtained, the relative frequencies of species A, B, and C are greater than or equal to π^* with γ confidence.

2.2.3 *A Priori* Sample Size With Full Pilot Data

Suppose a pilot study has been done in an area and that now the main study is to be performed. Assume that, using data from the pilot study, relative frequencies can be estimated for all observed species. If the total number of species, denoted by k, is unknown, see Bunge and Fitzpatrick (1993) for a list of papers which might be applied to estimate k. If k is estimated, one can hypothesize that the additional species not observed during the pilot study occur with relative frequency equal to (or less than) $1/n_p$, where n_p is the pilot study sample size. Suppose further that interest is in obtaining y out of k species with γ probability. No specification is given as to *which* particular set of y species are desired, only that y of k be obtained.

Let $m = k - y$ represent the number of species which can be missed. Let $\pi_{(m+1)}$ represent the (estimated or known) relative frequency of the $(m+1)$-st rarest species. Unless sample size is very large and precision is correspondingly high, estimated relative frequencies which are "close" to one another probably should be treated as equal due to the statistical error present in the estimates. To make "close" more rigorous, we define a "window" of estimated relative frequency values (henceforth referred to

as "the window"), inside of which frequencies are treated as exactly equal. Aside from being slightly artificial, defining such a window vastly improves the performance of the approximation. Nothing in the mathematics of our situation dictates an optimum window size. We (and the reader) are free to choose a window which works well in simulation and case studies. We have found that treating all estimated relative frequencies inside a window of $\pi_{(m+1)} \pm 0.9\pi_{(m+1)}$ as equal yields good approximate sample sizes and is the window we will use in this example.

Having defined a window, let r^* be the number of relative frequencies inside the window, let π^* be the average of all relative frequencies inside the window (divisor in this average is r^*), and let r_b represent the number of relative frequencies less than the lower endpoint of the window interval. Assuming that individuals are encountered in random order, the (approximate) smallest sample size such that y of k species are obtained with γ confidence is,

$$n \approx \frac{1}{b} \frac{ln(1-\gamma^a)}{ln(1-\pi^*)} \tag{2.2}$$

where $b = k + 1 - y - r_b = m + 1 - r_b$, and $a = \binom{r^*}{\mathrm{b}}^{-1} = \frac{\mathrm{b!}(r^*-\mathrm{b})!}{r^*!}$. Equation 2.2 will be called the ratio approximation.

As an example, consider the pilot data in Table 2.1. Table 2.1 contains the estimated relative frequencies of 31 taxa of *macrobenthos* collected in a pilot study of Beaver Creek, Oregon in 1994. "Taxa" are taxonomic categories of less resolution than species. For our purposes taxa are equivalent to species, but we will continue to use the proper term, taxa, for the remainder of this example.

In the columns labeled *Taxa #* of Table 2.1, each taxa name has been replaced by a number as follows: the rarest taxa are assigned the smallest numbers (1 through 6), the next rarest taxa are assigned the next smallest numbers (7 and 8), and so on until the most common taxa is assigned the largest number (31). Under the heading *Original* in Table 2.1, the *Estimated Relative Frequency* columns contain the fraction of sampled individuals which belonged to each of the 31 taxa found during the pilot study.

Original				Reassigned	
Taxa #	Relative Frequency(%)	Taxa #	Relative Frequency(%)	Taxa #	Relative Frequency(%)
1,2,3,4,5,6	0.032	21	0.810	1,2,3,4,5,6	0.032
7,8	0.065	22	0.842	7,8	0.065
9,10	0.097	23	0.907	9,10,...,24	0.463
11,12	0.130	24	0.939	25	1.328
13	0.162	25	1.328	26	1.814
14	0.227	26	1.814	27	5.602
15	0.259	27	5.602	28	7.902
16	0.356	28	7.902	29	8.938
17	0.453	29	8.938	30	9.618
18	0.583	30	9.618	31	57.059
19	0.680	31	57.059		
20	0.745				

Table 2.1: Original and reassigned relative frequencies of *macrobenthos* in Beaver Creek, Oregon ordered from rarest to most common. *Reassigned* relative frequencies are original relative frequencies except that the 16 frequencies in the interval [0.068%,1.29%] are assigned their average.

Suppose for the main study, researchers are interested in obtaining at least 13 taxa with 90% confidence. In this example k=31. Given the goal of obtaining 13 taxa with 90% confidence, y=13 and γ=90%, which implies m=31–13=18, and $\pi_{(m+1)}=\pi_{(19)}$ =0.68%. The window interval is $[\pi_{(m+1)}(0.1), \pi_{(m+1)}(1.9)]$ = [0.068%, 1.29%] and results in π^*=0.463%, r^*=16, and r_b=8. Replacing the original estimated relative frequencies which fell inside the window by π^* results in the estimated relative frequencies under the heading *Reassigned* in Table 2.1.

Applying the ratio approximation with $b = 18 + 1 - 8 = 11$ and $a = 1/4368 = 0.0002289$, we estimate that a sample size of approximately 208 individual *macrobenthos* is required in order to have 90% confidence that 13 or more of the 31 taxa will

be obtained. To be accurate, researchers must assure that individual macrobenthos are encountered in random order from the population.

How do we assess the accuracy of the estimate of 208 individuals? It seems reasonable to hypothesize that the estimated relative frequencies from the pilot study are true and calculate an "exact" sample size using these hypothesized numbers. Unfortunately, due to the large number of taxa in this real example (31), computer resources (time and memory) were too great and we could not compute this exact sample size for comparison. However, we were able to compute another approximation, the Monte Carlo approximation (see Section 2.3.2), which we know to be very close to the true value. The Monte Carlo approximation yielded a sample size of 163±2 meaning that the exact sample size is somewhere between 161 and 165. Taking 163 as the exact sample size, we see that the ratio approximation (i.e., n=208) is 28% too large.

2.3 Sample Size Methods

In this section we describe exact calculation of sample size required to obtain y species with γ confidence, and we derive our two approximations. In what follows, we will use the term "species" as a label for the groups in a population. Our methods remain equally applicable, however, whenever any type of group is defined in a population. We also refer to a "draw" as the act of obtaining and processing a single member from the population. Examples of "draws" include identifying the taxa of a single insect, catching a fish in a net and identifying its species, or inspecting a single component in an assembly line process. Additionally, we will dispense with the term "relative frequency" as it was used in Section 2.2 to represent the probability of obtaining a member of a species on any given draw and simply call this quantity a "probability". In addition to the notation of Section 2.2, we use the symbol n to represent sample size and the symbol $\boldsymbol{\pi}$ to represent the vector of species probabilities. That is, we let $\boldsymbol{\pi}$ contain the elements $\pi_{(1)}, \pi_{(2)}, \ldots, \pi_{(k)}$ where $\pi_{(i)}$ is the probability

of obtaining a member of species i on any draw (note $\pi_{(1)} + \pi_{(2)} + \ldots + \pi_{(k)} \equiv 1$). For convenience, we order the members of $\boldsymbol{\pi}$ so that $\pi_{(1)} \leq \pi_{(2)} \leq \ldots \leq \pi_{(k)}$.

2.3.1 Exact Sample Size

Appendix A presents a probability function which we use to model our situation. If we let $\mathbf{Y}$ represent the (random) number of species obtained at the end of n draws, the probability function in Equation A.4 (page 78) gives the probability that $\mathbf{Y}$ is less than or equal to a specified value under the assumption that each draw is independent.

Our goal is to calculate the smallest sample size, n, such that $Pr(\mathbf{Y} \geq k - m \mid n, \boldsymbol{\pi}) \geq \gamma$ where the distribution of $\mathbf{Y}$ is given by Equation A.4. Determination of such an n can be accomplished by setting one minus Equation A.4 equal to γ and solving for n iteratively. The well-known Newton-Raphson iterative solution algorithm has worked well for us here. Alternatively, one minus Equation A.4 can be computed for a range of n values and linear interpolation used between two successive values of n to achieve γ probability if necessary. $Pr(\mathbf{Y} \geq k - m \mid n, \boldsymbol{\pi})$ as a function of n is nearly linear when the minimum value in $\boldsymbol{\pi}$ is less than 0.05 and/or n is large. However, linear interpolation does not guarantee *the* exact solution to $Pr(\mathbf{Y} \geq k-m \mid n, \boldsymbol{\pi}) \geq \gamma$ is found.

Why are approximations needed if an exact value can be obtained through iteration? In many cases, an exact answer is not available because computations associated with Equation A.4 cannot be carried out in a reasonable amount of time. Equation A.2 in the appendix requires enumeration of all $2^k - 1$ possible (non-empty) subsets of the integers 1,...,k. When k=50 and assuming that a computer is available which can perform one million floating point operations per second, we estimate that it would take roughly nine centuries to compute a single point in the distribution of $\mathbf{Y}$. Even if a machine could perform one quadrillion (10^{15}) floating point operations per second, we estimate it would take a minimum of 20 million centuries to compute a

single point in the distribution of $\mathbf{Y}$ when k=100, provided adequate memory was available.

2.3.2 Monte Carlo Approximation

The Monte Carlo approximation uses an algorithm which parallels the exact solution in that it solves for sample size iteratively, but differs from the exact technique in that it uses a Monte Carlo generation technique to approximate points in the distribution of $\mathbf{Y}$. Our Monte Carlo approximation to $Pr(\mathbf{Y} \leq y \mid n, \boldsymbol{\pi})$ is motivated by Equation A.1 in the appendix. Steps in approximating Equation A.4 are outlined below. Given a vector $\boldsymbol{\pi}$ and sample size n, the steps in approximating Equation A.4 are:

1. Generate n random integers such that each has probability $\pi_{(i)}$ of equaling i.
2. Compute the number of distinct integers among the n generated. Call this number y_r.
3. Store y_r as a single realization of the random variable $\mathbf{Y}$.
4. Repeat steps 1 through 3 a minimum of 10,000, preferably 20,000, times. Call the number of repeats n_{iters}.
5. Approximate $Pr(\mathbf{Y} \leq y \mid n, \boldsymbol{\pi})$ as the proportion of y_r less than or equal to y (i.e., $\sum_{y_r} I(y_r \leq y)/n_{iters}$).

If θ is the true probability that $\mathbf{Y}$ equals y given n and $\boldsymbol{\pi}$, then the variance of our Monte Carlo estimate of $Pr(\mathbf{Y} \leq y)$ is $\theta(1-\theta)/n_{iters}$. If θ is 0.90, our Monte Carlo estimate is somewhere between 0.894 and 0.906 roughly 95% of the times we compute it when 10,000 iterations are performed. The endpoints 0.894 and 0.906 are θ plus and minus twice the estimate's standard error of 0.006 when n_{iter}=10,000.

Computer code, written in S+, to compute our Monte Carlo approximation to the distribution of $\mathbf{Y}$ given $\boldsymbol{\pi}$ and n appears in Table 2.2. There are a number

```
F.cdf <- function( n, p ){
        iters <- 10000; y <- rep(0,length(p))
        for(i in 1:iters){
                yy <- sum( !duplicated( sample(length(p), n, T, p )))
                y[yy] <- y[yy] + 1 }
        return( cumsum(y)/ iters ) }
```

Table 2.2: S+ code for Monte Carlo approximation to the distribution of $\mathbf{Y}$.

of improvements to the code in Table 2.2 which both speed calculations and make wiser use of memory, but require more statements and in the interest of space are not presented. After creation of the function in Table 2.2, the S+ statement which approximates $Pr(\mathbf{Y} \leq y \mid n, \boldsymbol{\pi})$ is `F.cdf(n,pi)[y]`.

The Monte Carlo approximation is highly accurate and computationally faster than the exact method, but still requires a significant amount of time to compute. In the next section, we develop a closed form approximation which can be computed on most handheld calculators.

2.3.3 Ratio Approximation

We attempt to motivate our ratio approximation and then present its formal derivation.

Following notation of previous sections, suppose the smallest probability in $\boldsymbol{\pi}$ is much smaller than the next smallest probability. The smallest probability in $\boldsymbol{\pi}$ is $\pi_{(1)}$, the next smallest is $\pi_{(2)}$. If all species are desired, it turns out that we can essentially ignore all species except the one with smallest probability and focus on obtaining a member of this smallest species. In doing so, we reduce the problem from a multinomial situation to a binomial situation and a closed form for sample size can

be worked out. If there exist several species with probability "close" to the smallest (say, $\pi_{(3)} \approx \pi_{(2)} \approx \pi_{(1)}$), reduction of the problem from multinomial to binomial is too rough. In most situations, the approximation can be improved by defining a "window" contained in the interval [0,1] inside which probabilities are treated as equal and computations adjusted accordingly.

We now derive our ratio approximation. Let $\boldsymbol{\pi}$, m, and γ be given. Recall that m= k–y, the number of species which are not required. Define a "window" contained in [0,1] as $[\delta_L,\delta_U]$ where $\delta_L = max((1-\delta)\pi_{(m+1)}, 0)$ and $\delta_U = min((1+\delta)\pi_{(m+1)}, 1)$. The nonnegative number δ determines width of the window and is set by researchers familiar with the population under study. In our simulations and example populations we have found $\delta < 1$ to be most useful and, in the absence of other information, suggest setting δ=0.90. Now let π^* be the average of all probabilities inside the window, r^* be the number of probabilities inside the window, and r_b be the number of probabilities less than the lower endpoint of the window. To summarize our notation:

$\boldsymbol{\pi}$ = hypothesized vector of species probabilities (size k by 1)

m = number of species which can be missed (i.e., it is desired to obtain members from y=k–m species)

γ = desired confidence in obtaining k–m species

δ = parameter determining width of probability window

δ_L = lower endpoint of window = $max((1-\delta)\pi_{(m+1)}, 0)$

δ_U = upper endpoint of window = $min((1+\delta)\pi_{(m+1)}, 1)$

r^* = number of probabilities inside window = $\sum_i I(\delta_L \leq \pi_{(i)} \leq \delta_U)$

r_b = number of probabilities below window = $\sum_i I(\pi_{(i)} < \delta_L)$

π^* = average probability in window = $\sum_i \pi_{(i)} I(\delta_L \leq \pi_{(i)} \leq \delta_U)$ / r^*

We wish to approximate the smallest n such that $Pr(\mathbf{Y} \geq k - m \mid \boldsymbol{\pi}, n) \geq \gamma$. We do this by ignoring the r_b species with probability less than δ_L and focus on

obtaining members from $(r^* + r_b - m)$ species among the r^* species with probabilities between δ_L and δ_U. In doing so, we will likely have already obtained members from the $k-(r^*+r_b)$ species with probabilities above the window and thus have $(k-(r^*+r_b))+(r^*+r_b-m) = k-m$ species altogether.

The probability that any single one of the r^* species with probability between δ_L and δ_U has a member at the end of sampling is approximately $1-(1-\pi^*)^n$ (the count of any single species has a binomial$(n,\pi_{(i)})$ marginal distribution). If we regard obtaining a member from each of these r^* classes as r^* (at least approximately) independent Bernoulli random variables each with probability $1-(1-\pi^*)^n$, then their sum (the total number of the r^* with a member) is at least approximately a binomial random variable with parameters r^* and $1-(1-\pi^*)^n$. Thus we approximate

$$Pr(\mathbf{Y} \geq k-m \mid n, \boldsymbol{\pi}) \approx Pr(\mathbf{X} \geq r^* + r_b - m \mid r^*, 1-(1-\pi^*)^n)$$

where $\mathbf{X}\sim$ binomial$(r^*,1-(1-\pi^*)^n)$, and find the smallest n such that the right-hand side is greater than γ.

We now seek a closed-form approximation to the binomial cumulative distribution function of $\mathbf{X}$. If we find such an approximation, we can set it equal to 1–γ and solve for n.

Let $\mathbf{B} \sim$ binomial(n,θ). If θ (=$1-(1-\pi^*)^n$ in our case) is small then for b close to n,

$$\begin{aligned} Pr(\mathbf{B} \leq b) &= 1 - Pr(\mathbf{B} \geq b+1) \approx 1 - Pr(\mathbf{B} = b+1) \\ &= 1 - \binom{n}{b+1} \theta^{b+1}(1-\theta)^{n-b-1} \\ &\approx 1 - \binom{n}{b+1} \theta^{b+1} \\ &\approx exp\left\{ -\binom{n}{b+1} \theta^{b+1} \right\}. \end{aligned}$$

The first "$\approx$" follows because θ is small, the second "$\approx$" because $1-\theta \approx 1$, and the third "$\approx$" uses the fact that $1-\psi \approx e^{-\psi}$ for ψ close to 0.

Transforming from "successes" to "failures" leads us to the approximation,

$$\begin{aligned} & Pr(\mathbf{X} \geq r^* + r_b - m \mid r^*, 1-(1-\pi^*)^n) \\ & = Pr(r^* - \mathbf{X} \leq m - r_b \mid r^*, (1-\pi^*)^n) \\ & \approx exp\left\{ -\binom{r^*}{m+1-r_b} (1-\pi^*)^{n(m+1-r_b)} \right\}. \end{aligned} \tag{2.3}$$

Setting Equation 2.3 equal to γ and solving for n yields,

$$n \approx \frac{1}{b} \frac{ln(-ln(\gamma^a))}{ln(1-\pi^*)}$$

where $b = m + 1 - r_b$ and $a = \binom{r^*}{b}^{-1}$. To further simplify the result, we use the fact that for ψ close to 1, $ln(\psi) \approx \psi - 1$ and substitute $1 - \gamma^a$ for $-ln(\gamma^a)$ and obtain Equation 2.2.

Although rough, this approximation performs well in a number of situations and has the added benefit of being easy to compute. As expected, Equation 2.2 performs better when $1-(1-\pi^*)^n$ is close to zero. When all species are desired (i.e., m=0), Equation 2.2 with δ=0 is highly accurate. In this case the approximation is $n \approx ln(1-\gamma)/ln(1-\pi_{(1)})$. The approximation performs reasonably well even when $1-(1-\pi^*)^n$ is not close to zero and m is large. Consider the example of Section 2.2.3 where $1-(1-\pi^*)^n = 1-(1-0.00463)^{208} = 0.619$, which is not close to zero, and m=18, yet the relative error in the approximation is only 28%.

2.4 Performance of Ratio Approximation

In ecology and biology, many populations contain a few "common" species and a large number of "rare" species which are present but rarely seen. The data from Beaver Creek in Table 2.1 are a good example of this phenomenon. Not surprisingly, the "rare" species drive sample size considerations when obtaining species is a goal. We would hope the ratio approximation works well in these situations.

2.4.1 Simulation

We investigated the accuracy of our ratio approximation when the underlying probabilities (values in $\boldsymbol{\pi}$) are generated by a stochastic (random) model. Under this stochastic model, the $\pi_{(i)}$ are (scaled) exponential random variables and are repeated a random number of times.

Each iteration in our simulation involved the following steps:

1. **Generate a random probability vector ($\boldsymbol{\pi}$)**

   (a) Generate five random deviates from an exponential distribution whose mean is 1. Sort in *ascending* order. Call these numbers $\pi_{d_{(i)}}$ where $\pi_{d_{(1)}}$ is the minimum and $\pi_{d_{(5)}}$ is the maximum.

   (b) Generate another set of five random deviates from an exponential distribution whose mean is 1. Sort in *descending* order. Call these numbers r_i where r_1 is the maximum and r_5 is the minimum.

   (c) Scale the r_i and round so that their sum is 10.

   (d) Replicate $\pi_{d_{(i)}}$, in $\boldsymbol{\pi}$, r_i times.

   (e) Scale the 10 numbers in $\boldsymbol{\pi}$ so that they sum to 1.

2. **Compute exact and approximate sample sizes given $\boldsymbol{\pi}$ and requiring 2, 3, ..., 10 species.**

3. **Calculate percent difference as %Difference = (Approximate n - Exact n) / (Exact n) and store.**

Two hundred and fifty iterations of the above process were performed and results are displayed in Figure 2.1. In Figure 2.1, the desired confidence (γ) in obtaining each required number of species was 90% and window size (δ) was 0 (left panel) and 0.90 (right panel). Separate probabilities were generated for each panel of Figure 2.1 resulting in different sets of exact sample sizes. Relative frequency distributions (histograms) of the generated probabilities appear in the upper right corner of each

panel. For each required number of species (each integer on horizontal axis) the minimum, 10th, 20th, 30th, 50th, 70th, 80th, 90th, and maximum percentiles of the corresponding 250 percent differences were computed. Shaded regions in Figure 2.1 connect these percentiles and are labeled with the proportion of differences falling inside each region. For example, with $\delta = 0.90$ (right panel) and requiring 9 out of 10 species, 80% of the approximate sample sizes were within ±5% of exact sample size while all approximate sample sizes in this case were within ±10%. Median percent difference for each required number of species is represented by a white line in the 40% region. For reference, the mean and standard deviation of each set of 250 exact sample sizes are listed just above the horizontal axes. Thus, for example, with δ=0.90 and requiring all species (10), 80% of the approximate values were between 0% and -15% of the exact value, representing, on average, that 80% of the approximations were less than 72 (=477*0.15) units below their exact counterparts, even though some of the exact values were as high as 5000.

Inspection of Figure 2.1 reveals that the ratio approximation mimicked exact sample size in a useful way. The large percent differences displayed in Figure 2.1 are generally not worrisome because they occurred when exact sample sizes were small and where errors are likely to be inconsequential. For example, the maximum difference of 75% displayed in the right hand panel of Figure 2.1 occurred when 3 out of 10 species were required and represents, on average, an approximation of 10 for an exact value of 6. Unless sample units are expensive, such errors are likely not to matter. We prefer high accuracy when sample size is large and costs are proportionally higher.

As measured by percent difference, the ratio approximation improved when exact sample sizes got large (and highly variable). When requiring 8, 9, or 10 species, 80% of our approximations with $\delta = 0.90$ were within 13% of exact sample size and the majority (60%) were within 9%. In general, when requiring 8, 9, or 10 classes, exact sample sizes climbed above 100 and at times increased above 4000. Also, our approximation with either $\delta = 0$ or $\delta = 0.90$ was above exact values in most (60% or more) of the cases and consequentially provides a conservative sample size estimate in the sense that true confidence is greater than γ. The ratio approximation displays

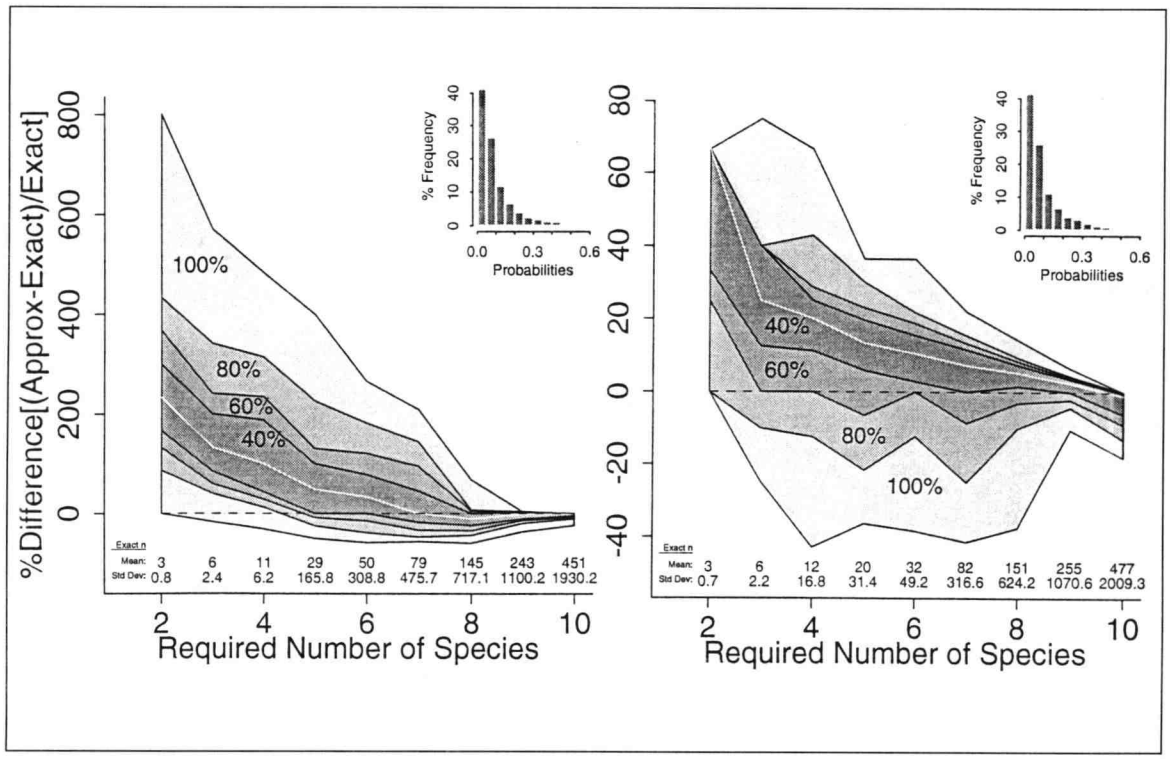

Figure 2.1: Results of the stochastic probability simulation comparing exact and approximate sample sizes with γ=90% and δ=0 (left panel) and δ=0.90 (right panel). Shaded regions are labeled with fraction of percent differences (of 250) inside each. White line is median percent difference. Small numbers above horizontal axis are mean and standard deviation of 250 exact sample sizes at each required number of species.

the desirable behavior of increased accuracy, as measured by percent difference, when sample sizes are large.

Comparing the two panels in Figure 2.1 we can conclude that a window width of 0.90 gives much better approximations than a window of width 0 when underlying probabilities display the particular structure generated in the simulation (i.e., few "common" species and many "rare" species). If plotted on the same graph, almost all shaded regions in the right-hand panel fall below the white median line in the left-hand panel indicating that nearly all approximations with $\delta = 0.90$ were better than 50% of the approximations with $\delta = 0$. Not shown in Figure 2.1 are approximate sample sizes computed using δ=0.75. Approximations using δ=0.75 were similar to those with δ=0.90 and separate presentation was not warranted.

2.4.2 More Species ($k > 30$)

Many populations in biology and ecology contain a large number (> 30) of species. We have seen examples from entomology where populations contain 80, 100, and even 1000 or more species. Unfortunately, simulations presented in the previous section could not be extended to these cases due to the excessive amount of time required to compute exact sample sizes.

In fact, computation of an exact sample size with $\boldsymbol{\pi}$ fixed and $k > 30$ eluded us. When $k > 30$, computer time and memory requirements were too great for the machines we had access to. The nature of the problem is such that time and memory requirements will eventually be too great for any computer of the type in existence today.

In order to investigate the ratio approximation when k is large, we use the Monte Carlo approximation as the "exact" sample size and compare the ratio approximation to the Monte Carlo approximation. We feel justified in doing so because of the nature of the Monte Carlo approximation. The Monte Carlo generation scheme mimics the model we are using in such a way that, at least in theory, we can compute a estimate of any point in the probability function of $\mathbf{Y}$ which is arbitrarily close to the corresponding exact value. This is a heuristic justification for stating that Monte Carlo approximate sample sizes converges to the corresponding exact sample size. The rate at which the Monte Carlo approximation converges to the exact value is unknown, but in the cases we could check ($k < 30$), the Monte Carlo approximation with 10,000 iterations was within 6 units of exact sample size in all cases. Sample sizes reported as "exact" below should be read as "exact ± 6".

The Monte Carlo approximation itself is time consuming to compute. We estimate that a single simulation of the type mentioned in the previous section, using the Monte Carlo approximation as the "exact" sample size, would require 52 days to run on a modern Unix workstation when k=50. We therefore abandon the random probability model and construct elements of $\boldsymbol{\pi}$ in a deterministic way.

As mentioned before, we have observed exponentially decaying probabilities in many natural populations and we wished to investigate the ratio approximation in this case. Thus, given a value of k, components of $\boldsymbol{\pi}$ were calculated as $\pi_{(i)} = e^{-i\beta} / \sum_{j=1}^{k} e^{-j\beta}$ for $i = 1, \ldots, k$ and β some positive constant. β was chosen so that the smallest component of $\boldsymbol{\pi}$ was in the neighborhood of 0.002. With the smallest probability in the neighborhood of 0.002, sample sizes were generally under 2000 and computation of the Monte Carlo approximation was feasible.

Figure 2.2 contains plots summarizing two cases. One case has $k = 20$ (left two panels), the other has $k = 50$ (right two panels). The desired confidence in both cases was $\gamma = 90\%$. In reality, we made similar computations for values of k between 10 and 50 in steps of 5 but found that all results were similar in character to the cases of $k = 20$ and $k = 50$ and chose to present these two cases only.

The top two panels of Figure 2.2 present graphs of sample sizes computed four ways: the Monte Carlo approximation, Equation 2.2 with a window size of 0, Equation 2.2 with a window size of 0.75, and Equation 2.2 with a window size of 0.90. The lower two panels of Figure 2.2 present graphs of percent difference for each of the approximations, taking the Monte Carlo approximate sample size as exact.

Three conclusions are drawn from these computations. First, approximate sample sizes computed using a window size of 0.75 are not substantially different than those computed using a window size of 0.90. This conclusion was noted in the stochastic probability example above and is substantiated by the closeness of the two lines in all panels of Figure 2.2. Second, the ratio approximation which uses a 0.75 or 0.90 window is better than the approximation using no window. This conclusion was also noted in the stochastic probability example and is substantiated in all panels of Figure 2.2 where lines for window widths 0.75 and 0.90 are fundamentally different and closer to exact sample size than lines for window width 0. Third, the ratio approximation (with non-zero window size) continues to perform well when the number of classes increases to 50. When requiring half or more of the classes to be represented, the error in our approximation was less than 25% when $k = 20$, and under 50% when

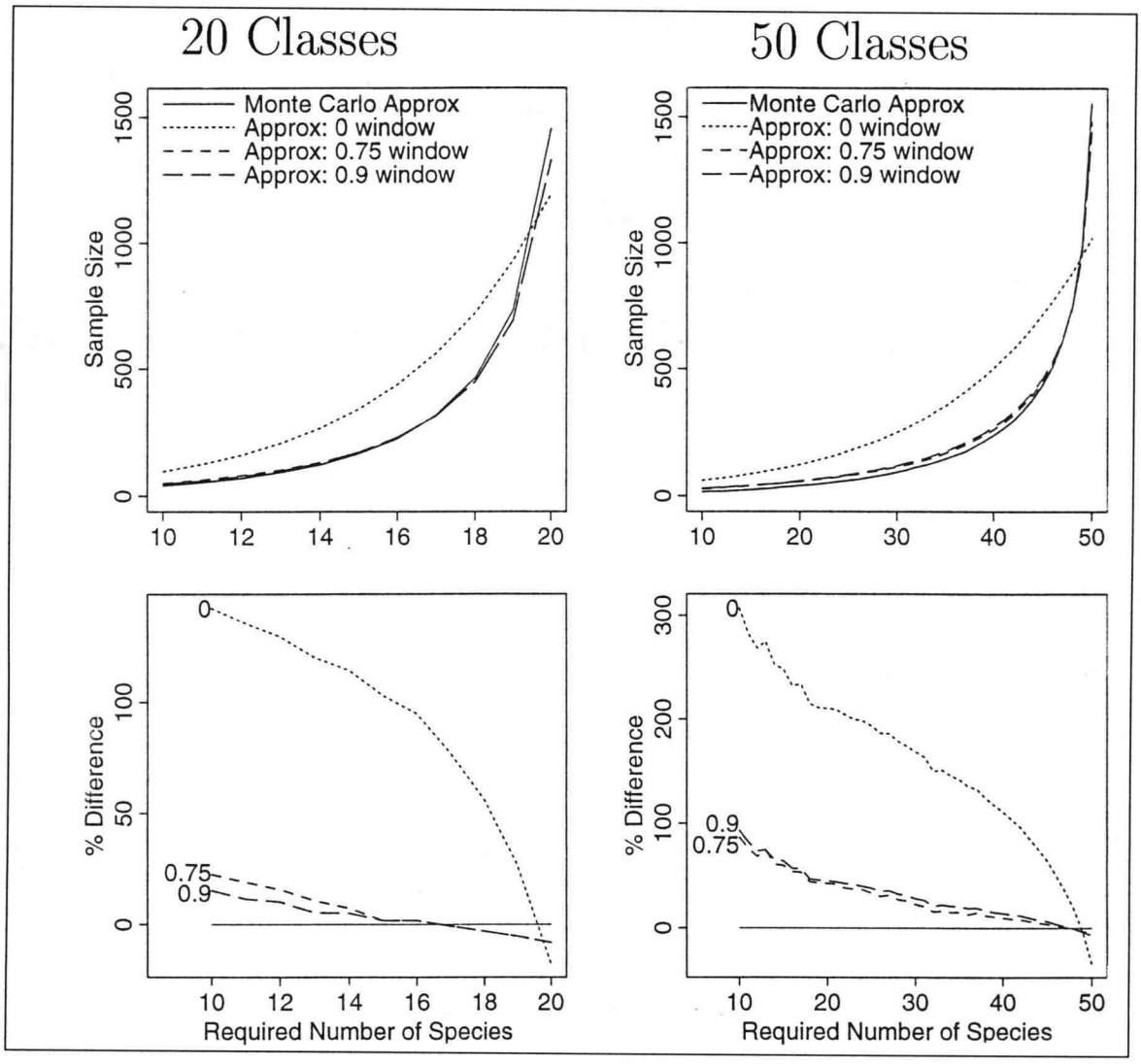

Figure 2.2: Comparing exact and approximate sample sizes for a large number of species and fixed probability vector. Left panels are for $k = 20$, right panels are for $k = 50$. Percent difference for each approximation is (Approximate - Monte Carlo approx) / (Monte Carlo approx). Value of the γ parameter is 90%. Note scale differences in vertical axes.

$k = 50$. When requiring all or nearly all classes, the error in our approximation drops to about 10% for both $k = 20$ and $k = 50$.

Noting the axis scale differences between the two lower panels of Figure 2.2 suggests that the relative error of the ratio approximation for a fixed required number of species gets progressively larger when k increases.

2.5 Discussion of Assumptions

All results in this paper, including computation of exact sample sizes, make use of the multinomial distribution as an underlying model. While every theoretical model fails, in some way, to exactly mimic all features of the real systems it attempts to represent, the multinomial is a reasonable model in a wide rage of applications and can provide insight into reasonable actions in many situations. Strictly speaking, however, the multinomial distribution requires both an infinite number of individuals in the population and that species membership of each individual be independent of the species membership of every other individual. Among these two assumptions, violating the independence assumption is more serious and is treated in the next paragraph. The other assumption is of little consequence when population size is large relative to sample size.

The independence assumption of the multinomial warrants some additional comments. In many populations, individuals of the same species tend to occur together, or in "clumps" (schooling fish and herding mammals are two examples). In order for the techniques outlined in this paper to mimic reality, every effort should be made to thoroughly "mix" the population of interest or otherwise insure that individuals are encountered at random. If the sampling mechanism collects entire "clumps" and no "mixing" of the "clumps" is done, the independence assumption has been violated. If the independence assumption is badly violated, the real sample size needed to obtain some number of species may be substantially larger than what our approximation and the multinomial predict. In general, the effects of "clumping" or non-independence on sample size are unknown and is an aspect of this problem in need of further research.

An alternative model which does not require the infinite population assumption is the multivariate hypergeometric distribution. Nath (1973) has results relevant to this case. As with the multinomial, computations involved in the multivariate hypergeometric distribution are complex and more research is needed into simplifying approximations for this case. If sample size is less than 15% of total population size,

the multinomial distribution of this paper is essentially equivalent to the multivariate hypergeometric and results herein can be applied without much concern.

At times, individuals from a population are available only through "batches" (distinct from "clumps" mentioned above) of fixed or random size. "Batches" could be jars of material, sweeps of a net, or sampled regions containing multiple individuals and is analogous to collecting bubble-gum cards wherein each package contains a fixed number of cards. If it is possible for more than one species to be present in a batch and individuals enter the batch at random, the results of this paper can be applied without concern. If, however, no two members of the same batch are of the same species, the independence assumption of the multinomial is violated. It is not surprising that the real required sample size in this case is less than what our approximations and the multinomial predict. If the average batch size is represented by $\bar{b}$ and our approximations or exact calculations indicate that a sample size of n is required, then approximately $n/\bar{b}$ batches are required.

2.6 Conclusions

In this paper we considered the question of how many samples to take in order to obtain $k-m$ classes with γ probability when a multinomial sample is taken from a population containing k classes. We described calculation of exact sample size required to achieve our goal and presented two approximations to this exact sample size, one termed the Monte Carlo approximation, the other termed the ratio approximation. Approximations are valuable here because often exact computations are involved and time consuming. If exact calculations cannot be carried out, choice of which approximation to use depends in large part on the speed with which an answer is needed. If adequate time exists, the Monte Carlo approximation is highly accurate and is the recommended technique. If an answer is needed immediately or a computer is not available, the ratio approximation provides a reasonable answer in almost all cases.

Chapter 3

Rank Methods Comparing Finite Distribution Functions Under General Designs

Trent L. McDonald and N. Scott Urquhart

Abstract

This paper explores testing for differences in finite population distribution functions which are defined on populations of equal size. The major goal of this paper is to expose the influence which sampling design has on the usual two-sample Wilcoxon rank sum test and to propose an alternate which is less influenced by designs. Focus is placed on performance of the Wilcoxon procedure and the proposed procedure under *randomized* general random sampling which includes simple random sampling, grouped random sampling, and probability proportional to an auxiliary variable sampling. The proposed test is found to have approximately correct nominal size under a wide range of general random samples while the Wilcoxon rank sum exhibited extreme violations in size in many cases. The proposed test is shown to have increasing power to detect both a shift and extension (change in variance) alternative hypothesis. Simple random sampling is shown to be a good design for testing purposes.

Keywords: population, survey, hypothesis testing, nonparametric

3.1 Introduction

Most, if not all, students attempting a first course in quantitative survey methods are exposed to techniques which estimate a population proportion. Cochran (1977), Särndal et al. (1992), Kish (1965) and just about every other book on quantitative finite population surveys describe estimation of a population proportion. All of these techniques which estimate proportions are fundamentally estimating a single point in a distribution function. Even estimation of the population median can be regarded as estimation of a quantile from a population distribution function (Särndal et al., 1992). Thus, distribution functions have been a part of finite population surveys for

some time and, surely, will continue to be a part of finite population statistics in the future.

Recently, some investigators have embraced explicit estimation of distribution functions as a fundamental design objective. The the National Stream Survey (Sale et al., 1988; Overton, 1985), the National Lake Survey (Linthurst et al., 1986; Overton et al., 1986; Overton, 1986), and the Environmental Monitoring and Assessment Program (EMAP) (Overton et al., 1990) all have implemented components or analysis plans which call for estimation and use of population distribution functions. Stehman and Overton (1994b) list distribution functions as important for population description in environmental sampling. Distribution functions provide a much more complete characterization of the population than do traditional parameters like the mean and total (Sedransk and Sedransk, 1979). Kuk (1988), Rao et al. (1990), and Rao (1994) give estimators for the entire distribution function. Särndal et al. (1992) give techniques to estimate any single point in a distribution function. Särndal et al. (1992) and Stehman and Overton (1994b) give techniques for placing approximate point-wise confidence intervals on estimated distribution functions.

Inevitably, users of estimated finite population distribution functions will wish to compare two or more distribution functions. Such comparisons are natural and can be insightful, especially if the two distribution functions under study differ in their tails. Such tail differences may reflect a difference in the number of "rare", "abnormal", or "damaged" elements. Traditional tests parallel to Student's t-test are fundamentally dependent upon differences in means and may not detect differences in the tails of distributions.

Given an interest in distribution functions, how should we compare two or more defined on separate finite populations? The comparison is especially difficult when samples are collected under non-standard designs such as probability proportional to size (πpx) or multi-stage cluster designs. If a researcher is fortunate enough to have data collected under simple random sampling, techniques borrowed from infinite population statistics (*i*ndependent and *i*dentically *d*istributed samples) can usually be applied without much concern. Many *i.i.d.* hypothesis tests are available which

test for differences in various characteristics of the underlying distribution functions. Student's t test (see, for example, Sokal and Rohlf (1981) and Zar (1974)) and the F test (Sokal and Rohlf, 1981) are parametric tests for differences in central tendency and variance. Wilcoxon's rank sum procedure (Wilcoxon, 1945; Mann and Whitney, 1947; Lehmann, 1975) is a nonparametric test for differences in two distribution functions. Levene's test (Levene, 1960) is a nonparametric test for differences in dispersion. Wilcoxon's test is particularly useful because it tests whether one variable is "stochastically larger" than another and has the ability to detect differences in the tails of distributions (Lehmann, 1975). If Wilcoxon's or similar test had predictable size and increasing power for increasingly disparate alternatives in a finite population setting, it would be a useful test.

However, is it appropriate to apply an infinite population technique to finite populations? Is it appropriate to apply Wilcoxon's rank sum test when data are collected in some manner other than simple random sampling? Unfortunately, there is little information in the literature on this point. Sedransk and Sedransk (1979) make a case for use of distribution functions in comparisons, but, regarding their test statistic $d_{hh'}$, state "Because of uncertainty about the distribution of $d_{hh'}$, formal probability statements would be tenuous and are not presented here. Moreover, properties of the common nonparametric test procedures are unknown (and extremely difficult to develop) when complex sampling designs are employed and sample sizes are small" (Sedransk and Sedransk, 1979, Section 4, Page 757). Särndal et al. (1992, Section 13.5) provide some guidance when distribution functions arising from combined strata in a simple stratified design are to be compared (see also Overton (1985)). Rao and Scott (1984) and Rao and Scott (1987) describe adjustments to regular χ^2 tests for multiway contingency tables which account for non-standard sampling designs. By partitioning the respective domains and observing counts in each, Rao and Scott's techniques could potentially be adapted to test for differences among distribution functions using a χ^2 analysis.

The remainder of this article focuses on testing for differences among two distribution functions when samples are drawn using a particular type of general random

sample and when the two populations are of equal size. This focus will not cover all possible survey situations, but will provide guidance in many and will broaden the base of techniques available to practicing statisticians. In particular, the regular *i.i.d.* version of Wilcoxon's test and the usual Wilcoxon test with a finite population correction are studied with the goal of determining when each is applicable and what, if anything, can go wrong if the tests are naively applied to samples from a general design. Another testing procedure based on a Horvitz-Thompson type correction of the Mann-Whitney U statistics is proposed to overcome some of the problems Wilcoxon's test has under general designs. It is assumed throughout that the two populations of interest are of the same size and independently sampled by fixed-size randomized generalized random sample (GRS) designs. The size (Type I error rate) of the Wilcoxon procedures and the proposed test are of primary concern and is assessed through computer simulation. In cases where the testing procedures have approximately the same size, power of each test to detect a shift and extension alternative hypothesis is investigated.

Obtaining a testing technique with proper size and adequate power under general designs is the crux of the problem presented in this paper. Focus will be restricted to a particular type of general design, randomized general random sampling (GRS) designs. Fixed size general random samples are a generalization of the *randomized* variable probability systematic (VPS) designs described in Brewer and Hanif (1983), Stehman and Overton (1994a), and Sunter (1986). Under randomized VPS designs, first order inclusion probabilities are proportional to an auxiliary variable. GRS samples remove this proportionality restriction and require only that the first order inclusion probabilities sum to n, the fixed sample size (inclusion probabilities must also be in the interval (0,1]). Given a set of first order inclusion probabilities, both GRS and VPS sampling randomly permute the population prior to drawing a systematic sample of population units. Close cousins to the randomized GRS and VPS sample designs are the *ordered* GRS and VPS sampling designs (Brewer and Hanif, 1983; Madow, 1949) under which the population is not randomized prior to sample selection. Although interesting, ordered GRS designs are beyond the scope of this paper. Several

commonly used designs are special cases of the randomized GRS design including simple random sampling (SRS), randomized VPS, and *grouped* GRS wherein every unit in each of several groups has equal inclusion probability. The National Stream Survey (Kaufmann et al., 1988) implemented a randomized VPS design where sampling was proportional to the watershed area of stream segments. The lakes component of the Environmental Monitoring and Assessment Program (EMAP) (Linthurst et al., 1986; Overton et al., 1986; Overton et al., 1990) implemented a grouped GRS design such that lakes within each of five size classes were sampled with equal probability. Probabilities in the lakes component of EMAP were set such that the expected number of lakes observed in each size class was approximately equal.

As is common in articles which investigate estimators under general finite population sampling designs, a simulation or empirical study is undertaken which provides at least some degree of justification for conclusions (see, e.g. Stehman and Overton (1994a)). Simulations are useful in these situations because of the complexity of the probabilistic mechanisms inherent in general variable probability designs (Stehman and Overton, 1994a). The simulations conducted in this paper are designed to encompass a wide range of designs but do not exhaust the set of all possible designs. The simulations cover GRS designs with varying levels of correlation between responses and inclusion probabilities, varying levels of inclusion probability variance, varying number of distinct inclusion probabilities, varying relative sample size, and three different population structures. Results should be viewed the same as results of mathematical theorems. Provided the set of assumptions under which the simulation or theorem is built are satisfied, the results follow. If assumptions are not satisfied, the results may or may not follow.

This paper is organized as follows: Section 3.2 defines notation used throughout the paper including matrix operators, matrix functions, finite population quantities, and the null hypothesis; Section 3.3 gives the testing procedures; Section 3.4 describes the simulation experiments; Section 3.5 contains the results of the simulation experiments; and Section 3.6 discusses applications.

3.2 Notation and Assumptions

Matrix notation will be used throughout this paper. Matrix notation simplifies many finite population quantities and allows parallels to be drawn between finite population results and standard (infinite population) linear model theory. In the following, J will be used to represent a column vector of 1's whose size should be obvious from the context. When the size of J needs to be made explicit, it will be subscripted with the size, as in J_k. In addition to the regular matrix operations of addition (+), multiplication (*), transposition ($'$), and inversion (A^{-1}), element-wise division (/) and Kroneker product ($\otimes$) are needed here. Let the notation A=$\{a_{ij}\}$ mean that the element in the i-th row and j-th column of matrix A is the number a_{ij}. If matrices A=$\{a_{ij}\}$ and B=$\{b_{ij}\}$ are the same size, then define A/B = $\{ a_{ij}/b_{ij} \}$. Use of the symbol "/" for both scalar division and element-wise matrix division should not cause confusion because scalars are simply 1×1 matrices. For matrices A and B, let A $\otimes$ B = $\{ a_{ij}B \}$ for all i and j, where $a_{ij}B$ is regular scalar-matrix multiplication. Note that if A is a×b and B is c×d, then A $\otimes$ B is ac×bd.

Two matrix functions are needed here. For a matrix A, define $vec(A)$ to be the *row-major order* column vector representation of A. For example, if A=$\left[\begin{smallmatrix} a & b \\ c & d \end{smallmatrix}\right]$, then $vec(A) = \left[\begin{smallmatrix} a \\ b \\ c \\ d \end{smallmatrix}\right]$. The other matrix function generalizes the scalar indicator function. Let X be a r×1 column vector, Y be a 1×c row vector, and define I(X<Y) to be a r×c matrix of 0's and 1's where the element in the i-th row and j-th column of I(X<Y) is a 1 if the i-th element of X is less than the j-th element of Y, and 0 otherwise. For example, if X=$\left[\begin{smallmatrix} 1 \\ 3 \\ 4 \end{smallmatrix}\right]$ and Y=$[2\ 4]$, then I(X<Y) = $\left[\begin{smallmatrix} 1 & 1 \\ 0 & 1 \\ 0 & 0 \end{smallmatrix}\right]$. Any element-wise comparison operation can be an argument to I() and the definition still makes sense. Consequently, I(X $\leq$ Y) and I(X = Y) are matrices which have a 1 in the (i,j)-th element if the corresponding element-wise comparison is true (either $x_i \leq y_j$ or $x_i = y_j$ respectively) and a 0 otherwise. In the special case when Y is a scalar constant, say equal to t, I(X< t) is the same size as X and J'I(X< t) counts the number of times an element in X is less than t.

Matrix and vector ordering will be critical in defining the tests below. Consequently, we need an explicit notational representation of matrix and vector order. Let X be a column vector of size N. Let P=$\{p_i\}$ be a column vector containing the integers 1, 2, ..., N in some order. The notation X[P] will represent a vector whose i-th element is the p_i-th element of X. For example if X=$\begin{bmatrix} 0 \\ 10 \\ 5 \end{bmatrix}$ and P=$\begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$, then X[P]=$\begin{bmatrix} 5 \\ 0 \\ 10 \end{bmatrix}$. This notation will provide a handy way to represent a sorted vector. Note that P=$\begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$ sorts X in the example just given. For matrices, the notion of element ordering will be similar to that for vectors except that it will be applied to the row and column dimension of the matrix. Thus, if X is a N×M matrix and X_j represents the j-th column of X, the notation X[P,Q] will represent a matrix whose j-th column is X_{q_j}[P]. That is, the i-th row of X[P,Q] is a reordered, according to Q, version of the p_i-th row of X and the j-th column of X[P,Q] is a reordered, according to P, version of the q_j-th column of X. For example, if X=$\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$, P=$\begin{bmatrix} 2 \\ 1 \\ 3 \end{bmatrix}$, and Q=$\begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$ then X[P,Q]=$\begin{bmatrix} f & d & e \\ c & a & b \\ i & g & h \end{bmatrix}$.

To define notation for the finite population, let $\mathcal{U}_1$ be a finite set of N discrete units labeled u_i, $i = 1, \ldots, N$. $\mathcal{U}_1$ will be called the first *universe*. Let $\mathcal{U}_2$ be a finite set of M discrete units labeled v_i, $i = 1, \ldots, M$. $\mathcal{U}_2$ will be called the second *universe*. For every unit in both universes, let y_u and y_v be real-valued response variables which are functions only of the units. It will be convenient to collect all possible responses from a universe into a vector. Let $\boldsymbol{Y_u}$ be a N×1 vector in which the i-th element is y_{u_i}. Similarly, let $\boldsymbol{Y_v}$ be a M×1 vector in which the i-th element is y_{v_i}. Samples are defined as subsets of their respective universes. Let the first sample be a subset of size n of $\mathcal{U}_1$ and represented by s_1. Let the second sample be a subset of size m of $\mathcal{U}_2$ and represented by s_2. Let $\boldsymbol{y}_{s_1}$ be the vector of responses, y_u, from those units in the first sample and $\boldsymbol{y}_{s_2}$ be the vector of responses, y_v, from units in the second sample. Under a given design, let s_1 and s_2 be drawn in such a way that π_u and π_v are the first order inclusion probabilities for units u and v from $\mathcal{U}_1$ and $\mathcal{U}_2$ respectively. Let π_{uu^*} and π_{vv^*} be the second order inclusion probabilities of the pair of units (u,u*) from $\mathcal{U}_1$ and (v,v*) from $\mathcal{U}_2$. It will be convenient to collect both the first and second order inclusion probabilities into vectors and matrices. Let $\boldsymbol{\pi_u}$ and

$\boldsymbol{\pi_v}$ represent, respectively, N×1 and M×1 vectors containing all first order inclusion probabilities for units from $\mathcal{U}_1$ and $\mathcal{U}_2$ under a given design. The order of elements in $\mathcal{U}_1$ and $\boldsymbol{\pi_u}$ is connected in that the first element in $\boldsymbol{\pi_u}$ is the inclusion probability of the first element in $\mathcal{U}_1$, and so on. Similarly for $\mathcal{U}_2$ and $\boldsymbol{\pi_v}$. Let $\boldsymbol{\pi_{uu^*}}$ and $\boldsymbol{\pi_{vv^*}}$ represent, respectively, N×N and M×M matrices where each off diagonal element is a second order inclusion probability for a pair of units from, respectively, $\mathcal{U}_1$ and $\mathcal{U}_2$, and the diagonal elements are the first order inclusion probabilities for units from their respective universes. Again, the order of $\mathcal{U}_1$ and $\boldsymbol{\pi_{uu^*}}$ are connected in that the (i,j)-th element of $\boldsymbol{\pi_{uu^*}}$ is the second order inclusion probability for the i-th and j-th unit of $\mathcal{U}_1$. Similarly for $\mathcal{U}_2$ and $\boldsymbol{\pi_{vv^*}}$. Let $\boldsymbol{\pi}_{s_1}$ and $\boldsymbol{\pi}_{s_2}$ be the n×1 and m×1 vectors of first order inclusion probabilities of those units observed in the sample (i.e., of $\boldsymbol{y}_{s_1}$ and $\boldsymbol{y}_{s_2}$ respectively), ordered accordingly. To illustrate this vector notation, note that for all fixed-size designs, $J'\boldsymbol{\pi_u}$= n, $J'\boldsymbol{\pi_v}$= m, and $J'\boldsymbol{\pi_{uu^*}}$=n$\boldsymbol{\pi_u}'$.

The population distribution functions are defined as, $F_u(t)= (J'J)^{-1}J'I(\boldsymbol{Y_u} \leq t)$ and $F_v(t)= (J'J)^{-1}J'I(\boldsymbol{Y_v} \leq t)$. These are the proportion of elements in $\mathcal{U}_1$ or $\mathcal{U}_2$ for which y_u or y_v is less than or equal to t. Define $\boldsymbol{Z_{uv}} = I(\boldsymbol{Y_u} < \boldsymbol{Y_v}) + 0.5I(\boldsymbol{Y_u} = \boldsymbol{Y_v})$. $\boldsymbol{Z_{uv}}$ is a fixed N×M *population* matrix whose elements are 0's, 0.5's, and 1's. We also define $\boldsymbol{z_{uv}} = I(\boldsymbol{y}_{s_1} < \boldsymbol{y}_{s_2}) + 0.5I(\boldsymbol{y}_{s_1} = \boldsymbol{y}_{s_2})$. $\boldsymbol{z_{uv}}$ is a n×m *sample* matrix of those elements in $\boldsymbol{Z_{uv}}$ which were actually observed. Note that $J'\boldsymbol{z_{uv}}$ is the usual Mann-Whitney U statistic. Define $\boldsymbol{\mathcal{Z}} = vec(\boldsymbol{Z_{uv}})$. $\boldsymbol{\mathcal{Z}}$ is the NM×1 row-major order vectorized version of $\boldsymbol{Z_{uv}}$.

Throughout this paper, the two samples, s_1 and s_2, are assumed to be independent in that Pr(u$\in s_1$ and v$\in s_2$) = $\pi_u\pi_v$.

Finally, the null hypothesis being tested is defined. The null hypothesis tested in this paper is $F_u(t)$=$F_v(t)$ for all values of t. This hypothesis requires the "jumps" or support of $F_u(t)$ and $F_v(t)$ to be the same and *requires* N=M (or at least N= kM, where k is an integer). Consequently, M will be required to equal N in the remainder of this paper and M will be dropped from the notation. This null hypothesis is appropriate for most environmental monitoring studies where the same population is visited at different points in time. This null hypothesis is also appropriate if, because

of frame errors, two populations could theoretically be the same size even though their frames are different sizes. If population sizes are unknown but assumed large, techniques designed to test this null hypothesis can likely be applied without much concern. If N is known to be very different from M this particular null hypothesis is not interesting (because it is not true) and the results of this paper should be applied with caution.

3.3 Methods: The Testing Procedures

The Horvitz-Thompson theorem (Horvitz and Thompson, 1952) and the Horvitz-Thompson estimator for a population total play a central role in the testing procedures proposed in Section 3.3.1 and Section 3.3.2. The usual single population Horvitz-Thompson theorem is not restated here. The interested reader is referred to Cassel et al. (1977) and Särndal et al. (1992). Below, the Horvitz-Thompson estimator for the sum of values in $\boldsymbol{Z_{uv}}$, the estimator's mean, and its variance, are given and used. These quantities are the basis for tests proposed in Section 3.3.1 and Section 3.3.2.

To derive inclusion probabilities for elements in $\boldsymbol{z_{uv}}$, note that every value in $\mathcal{Z}$ is a function of one response from $\mathcal{U}_1$ and one response from $\mathcal{U}_2$. The probability of any value in $\mathcal{Z}$ being included in $vec(\boldsymbol{z_{uv}})$ is the product of the probabilities that the corresponding sample units are included in s_1 and s_2. Regarding $\mathcal{Z}$ as a population, an inclusion probability vector for $\mathcal{Z}$ can be written in the same way that an inclusion probability vector can be written for $\mathcal{U}_1$ and $\mathcal{U}_2$. The first order inclusion probability vector for $\mathcal{Z}$ is $vec(\boldsymbol{\pi_u \pi_v}')$. The order of elements in $vec(\boldsymbol{\pi_u \pi_v}')$ is correct if the ordering of $\boldsymbol{\pi_u}$ and $\boldsymbol{\pi_v}$ is correct. Correct ordering means that the first element in $vec(\boldsymbol{\pi_u \pi_v}')$ is the inclusion probability of the first element in $\mathcal{Z}$, and so on.

The second order inclusion probabilities for any two values in $\mathcal{Z}$ are not complicated, but can be difficult conceptually. Any pair of values in $\mathcal{Z}$, say z_{uv} and $z_{u^*v^*}$, are dependent upon four units, two from $\mathcal{U}_1$ and two from $\mathcal{U}_2$. By independence of

the samples, the probability that both z_{uv} and $z_{u^*v^*}$ are included in $vec(\boldsymbol{z_{uv}})$ is,

$$\pi_{z_{uv}z_{u^*v^*}} = \begin{cases} \pi_u\pi_v & \text{if u=u}^* \text{ and v=v}^* \\ \pi_u\pi_{vv^*} & \text{if u=u}^* \text{ and v}\neq \text{ v}^* \\ \pi_{uu^*}\pi_v & \text{if u}\neq \text{ u}^* \text{ and v= v}^* \\ \pi_{uu^*}\pi_{vv^*} & \text{if u}\neq \text{ u}^* \text{ and v}\neq \text{ v}^* \end{cases}.$$

By considering all possible $\pi_{z_{uv}z_{u^*v^*}}$ and the order of $\boldsymbol{\pi_{uu^*}}$ and $\boldsymbol{\pi_{vv^*}}$, the second order inclusion probabilities matrix for $\boldsymbol{\mathcal{Z}}$ can be written as $\boldsymbol{\pi_{uu^*}} \otimes \boldsymbol{\pi_{vv^*}}$. Again, the order of $\boldsymbol{\pi_{uu^*}} \otimes \boldsymbol{\pi_{vv^*}}$ is correct if the ordering of $\boldsymbol{\pi_{uu^*}}$ and $\boldsymbol{\pi_{vv^*}}$ is correct. Correct ordering for $\boldsymbol{\pi_{uu^*}} \otimes \boldsymbol{\pi_{vv^*}}$ means that, for example, the value in position (1,2) of $\boldsymbol{\pi_{uu^*}} \otimes \boldsymbol{\pi_{vv^*}}$ is the second order inclusion probability for the first and second value in $\boldsymbol{\mathcal{Z}}$.

Drawing the samples s_1 and s_2 amounts to realizing a value of the random matrix $\boldsymbol{z_{uv}}$. By independence of the samples, the first order inclusion probability vector for $vec(\boldsymbol{z_{uv}})$ is $vec(\boldsymbol{\pi}_{s_1}\boldsymbol{\pi}_{s_2}')$. If $z_{uv}= \text{I}(y_u<y_v) + 0.5\text{I}(y_u=y_v)$ is an arbitrary element in $\boldsymbol{z_{uv}}$, the Horvitz-Thompson estimator (Horvitz and Thompson, 1952; Särndal et al., 1992; Cassel et al., 1977) for the total $J_N'\boldsymbol{Z_{uv}}J_N = J'\boldsymbol{\mathcal{Z}}$ is,

$$\hat{t}_z = \sum_{u\in s_1}\sum_{v\in s_2}\frac{z_{uv}}{\pi_u\pi_v} = J_n'(\boldsymbol{z_{uv}}/\boldsymbol{\pi}_{s_1}\boldsymbol{\pi}_{s_2}')J_m \tag{3.1}$$

By the Horvitz-Thompson theorem, $\hat{t}_z$ is unbiased for $J'\boldsymbol{\mathcal{Z}}$ and its variance is known. The expected value of $\hat{t}_z$ is,

$$E[\hat{t}_z] = J'\boldsymbol{\mathcal{Z}} = \left(N^2 + \frac{N(N+1)}{2}\right) - \sum_{u\in\mathcal{U}_1} R_{y_u} \tag{3.2}$$

where R_{y_u} is the rank of y_u in $\boldsymbol{Y_u} \cup \boldsymbol{Y_v}$, replacing ties with the average rank. This expression makes it clear that the expected value of $\hat{t}_z$ is a linear combination of $\sum_{u\in\mathcal{U}_1} R_{y_u}$, the Wilcoxon rank sum statistic computed as if both universes were the "samples". By the Horvitz-Thompson theorem, the variance of $\hat{t}_z$ is

$$V(\hat{t}_z) = \boldsymbol{\mathcal{Z}}'\left[\left(\frac{\boldsymbol{\pi_{uu^*}}}{\boldsymbol{\pi_u}\boldsymbol{\pi_u}'} \otimes \frac{\boldsymbol{\pi_{vv^*}}}{\boldsymbol{\pi_v}\boldsymbol{\pi_v}'}\right) - JJ'\right]\boldsymbol{\mathcal{Z}}. \tag{3.3}$$

Under some designs (e.g., simple random sampling) all second order inclusion probabilities are known. Under other designs, some or all second order inclusion

probabilities may not be known. Under most randomized GRS, the second order inclusion probabilities can be known but are not usually because they are computationally difficult and time consuming to compute. Overton (1985) proposed a closed form approximation to the second order inclusion probabilities of any randomized GRS. Empirical studies by Stehman and Overton (1994a) suggest that the second order inclusion probability approximation performs well in many situations and is used below. Overton's approximation to the second order inclusion probabilities of a GRS design is,

$$\pi_{uu^*} = \frac{2(n-1)\pi_u\pi_{u^*}}{2n - \pi_u - \pi_{u^*}}$$

3.3.1 The Naive and Corrected Wilcoxon

A naive approach to testing in a finite population setting completely ignores sampling probabilities (and the fact that the populations are finite) and tests the null hypothesis using the usual infinite population Wilcoxon rank sum procedure. This naive approach is equivalent to assuming simple random sampling regardless of the design actually used. Thus, the *naive* Wilcoxon procedure assumes data were collected under simple random sampling and proceeds with a regular Wilcoxon rank sum test. The *corrected* Wilcoxon assumes simple random sampling, but acknowledges the finiteness of the populations and corrects variance accordingly.

The Wilcoxon rank sum statistic is

$$R = \sum_{u \in s_1} r_{y_u}$$

where r_{y_u} is the rank of the response from unit u in s_1 when ranked in the set of responses from the composite sample $s_1 \cup s_2$, replacing tied responses with the average rank. If simple random sampling is actually performed,

$$R = \frac{n(2m+n+1)}{2} - \frac{nm}{N^2}\hat{t}_z \tag{3.4}$$

because π_u=n/N for all u and π_v=m/N for all v under simple random sampling. Under simple random sampling, R is a linear combination of the Horvitz-Thompson

estimator $\hat{t}_z$ and the expected value and variance of R are known due to this connection with $\hat{t}_z$. The expected value and variance of R under simple random sampling are,

$$E[R] = \left(\frac{n(n+1)}{2} - \frac{nm(N+1)}{2N}\right) + \frac{nm}{N^2}\sum_{u\in\mathcal{U}_1} R_{y_u} \tag{3.5}$$

$$V(R) = \left(\frac{nm}{N^2}\right)^2 V(\hat{t}_z) \tag{3.6}$$

where R_{y_u} is the rank of y_u in the composite universe $\boldsymbol{Y_u}\cup\boldsymbol{Y_v}$ (ties replaced by average rank) and $V(\hat{t}_z)$ is given in Equation 3.3. Equation 3.5 and Equation 3.6 hold regardless of the validity of the null hypothesis (provided simple random sampling is performed). If the null hypothesis is true, $\sum_{u\in\mathcal{U}_1} R_{y_u} = N(2N+1)/2$ and $E[\hat{t}_z] = NM/2$, so that under simple random sampling

$$E_{srs}[R] = \frac{n(m+n+1)}{2}.$$

The naive testing approach pretends that $E[R] = n(m+n+1)/2 = E_{srs}[R]$ regardless of the design actually implemented and that the variance of R is $V_{nv}(R) = nm(m+n+1)/12$. The naive test then compares $(R - E_{srs}[R])/\sqrt{V_{nv}(R)}$ to a quantile from the standard normal distribution.

Under simple random sampling, a correction to $V_{nv}(R)$ can be obtained from Equation 3.6 which accounts for the finite nature of the two populations of interest, sampling variability, and inherent ties induced by the null hypothesis. This correction is obtained by expressing Equation 3.6 as $V_{nv}(R)$ times a "correction factor". Writing Equation 3.6 in this way is useful but somewhat arbitrary because the true variance of R in a finite setting is fundamentally different from the permutation variance of R derived in an infinite population setting. Nonetheless, under simple random sampling and the null hypothesis, the correct variance of R is,

$$V_c(R) = \left(\frac{nm(m+n+1)}{12}\right)\left(1 - \frac{N-n-m+nm(2N-1)}{N(N-1)(n+m+1)}\right). \tag{3.7}$$

Some algebra is required to arrive at this expression.

The corrected Wilcoxon test procedure is similar to the naive in that it pretends $E[R] = E_{srs}[R]$ regardless of the design implemented. The procedures differ in that

the corrected test uses $V_c(R)$ instead of $V_{nv}(R)$ for the variance of R. The corrected procedure compares $(R - E_{srs}[R])/\sqrt{V_c(R)}$ to a quantile from the standard normal distribution.

3.3.2 Proposed Test

The test proposed in this section uses a test statistic which is an "approximately" unbiased ratio estimator for the proportion of values in $\boldsymbol{Z_{uv}}$ equal to zero plus one half the proportion of values in $\boldsymbol{Z_{uv}}$ equal to 0.5. Equivalently, the test statistic can be regarded as estimating the probability $[F_{y_u - y_v}(0^+) + F_{y_u - y_v}(0^-)]/2$, where $F_{y_u - y_v}(t)$ is the proportion of values in $\boldsymbol{Y_u}J' - J\boldsymbol{Y_v}'$ which are less than or equal to t. This section is organized as follows: The test statistic and its (approximate) expected value are given; The approximate variance of the test statistic is given; The approximate variance is used to obtain method-of-moment estimates for the parameters of a Beta distribution; Significance of the observed statistic is obtained by integrating the Beta with estimated parameters.

The proposed test procedure uses the test statistic

$$t_p = \frac{\sum\limits_{u \in s_1} \sum\limits_{v \in s_2} \frac{z_{uv}}{\pi_u \pi_v}}{\sum\limits_{u \in s_1} \sum\limits_{v \in s_2} \frac{1}{\pi_u \pi_v}} = \frac{\hat{t}_z}{\widehat{N}^2}. \tag{3.8}$$

This statistic is the Horvitz-Thompson estimator, $\hat{t}_z$, "scaled" by the estimated population size. If simple random sampling is used, $\widehat{N}^2 = N^2$ and by solving for $\hat{t}_z$ in Equation 3.4 and substituting into Equation 3.8, we see that,

$$t_p = \frac{1}{nm}\left(\frac{n(2m+n+1)}{2} - R\right).$$

Thus, if simple random sampling is actually performed, t_p and R form equivalent tests.

The statistic t_p is a ratio of random variables and Särndal et al. (1992) state that ratios of this form are "approximately unbiased" estimators of means. In this situation, the mean being "approximately" estimated is the average of all elements

in the vector $\mathcal{Z}$. This average, denoted as μ_z, can be computed as $\mu_z = (J'J)^{-1}J'\mathcal{Z}$ $= E[\hat{t}_z]/N^2$. By substituting Equation 3.2 for $E[\hat{t}_z]$, $E[t_p]$ can be expressed as

$$E[t_p] \approx \frac{3N+1}{2N} - \frac{1}{N^2}\sum_{u\in\mathcal{U}_1} R_{y_u} \tag{3.9}$$

where "$\approx$" is meant to acknowledge the slight bias inherent in the ratio estimator t_p.

Särndal et al. (1992) state that the bias inherent in these types of ratio estimators is negligible in many cases and that the estimators are superior in other ways to non-ratio estimators. The simulation study described in Section 3.4 supports this observation. In fact, the bias of t_p for μ_z under the null hypothesis in the simulation was estimated at less than ± 0.01 ($\pm 2\%$) with samples of size 5 from both universes and ± 0.007 ($\pm 1.4\%$) for samples sizes of 10.

Särndal et al. (1992, pp. 178, equation 5.6.6) give an expression for the approximate variance of t_p derived using a first-order Taylor series expansion of the ratio. It is more convenient to write Särndal et al.'s expression as a quadratic form. The approximate variance of t_p is,

$$\tilde{V}(t_p) = \left(\frac{1}{N^2}\right)^2 (\mathcal{Z} - \mu_z J)'(C - JJ')(\mathcal{Z} - \mu_z J) \tag{3.10}$$

where $C = (\boldsymbol{\pi_{uu^*}}/\boldsymbol{\pi_u}\boldsymbol{\pi_u}') \otimes (\boldsymbol{\pi_{vv^*}}/\boldsymbol{\pi_v}\boldsymbol{\pi_v}')$. The "$\sim$" above the V in $\tilde{V}(t_p)$ is to acknowledge the (1st order) Taylor series approximation.

In $\tilde{V}(t_p)$, μ_z is the average of elements in $\mathcal{Z}$ and thus the deviations, $(\mathcal{Z} - \mu_z J)$, sum to zero. Using this fact and expanding the multiplication,

$$\tilde{V}(t_p) = \left(\frac{1}{N^2}\right)^2 \left[\mathcal{Z}'C\mathcal{Z} - 2\mu_z J'C\mathcal{Z} + \mu_z^2 J'CJ\right]. \tag{3.11}$$

Each sum in Equation 3.11 will be computed in turn below.

The sum $J'CJ$ in $\tilde{V}(t_p)$ depends only on the design and is known if all first and second order inclusion probabilities are known. Assuming all first and second order inclusion probabilities are known and exploiting the Kroneker form of C,

$$J'CJ = (\bar{w}_u N + \bar{w}_{uu^*}N(N-1))(\bar{w}_v N + \bar{w}_{vv^*}N(N-1)), \tag{3.12}$$

where

$$\bar{w}_u = \frac{1}{N}\sum_{u\in\mathcal{U}_1}\frac{1}{\pi_u} \qquad \bar{w}_{uu^*} = \frac{1}{N(N-1)}\sum_{u\neq u^*}\sum\frac{\pi_{uu^*}}{\pi_u\pi_{u^*}}$$

$$\bar{w}_v = \frac{1}{N}\sum_{v\in\mathcal{U}_2}\frac{1}{\pi_v} \qquad \bar{w}_{vv^*} = \frac{1}{N(N-1)}\sum_{v\neq v^*}\sum\frac{\pi_{vv^*}}{\pi_v\pi_{v^*}}.$$

The numbers $\bar{w}_u$ and $\bar{w}_v$ are the average diagonal element in $\boldsymbol{\pi_{uu^*}/\pi_u\pi_u}'$ and $\boldsymbol{\pi_{vv^*}/\pi_v\pi_v}'$ respectively. The numbers $\bar{w}_{uu^*}$ and $\bar{w}_{vv^*}$ are the average off diagonal element in $\boldsymbol{\pi_{uu^*}/\pi_u\pi_u}'$ and $\boldsymbol{\pi_{vv^*}/\pi_v\pi_v}'$ respectively.

As stated before, under the null hypothesis, $E[\hat{t}_z] = N^2/2$ and consequently $\mu_z = E[\hat{t}_z]/N^2 = 1/2$. Under the null hypothesis and provided no ties occur within $\boldsymbol{Y_u}$ or $\boldsymbol{Y_v}$, $\boldsymbol{\mathcal{Z}}$ contains exactly $[N(N-1)/2]$ 1's, exactly $[N(N-1)/2]$ 0's, and $[N]$ 0.5's. If ties occur within $\boldsymbol{Y_u}$ or $\boldsymbol{Y_v}$, there are more than $[N]$ 0.5's in $\boldsymbol{\mathcal{Z}}$but the approximation assuming $[N]$ 0.5's is still good because the sum $J'C\boldsymbol{\mathcal{Z}}$ is relatively unchanged in this case. Because approximately half the elements in $\boldsymbol{\mathcal{Z}}$ are 1, the sum $J'C\boldsymbol{\mathcal{Z}}$ is approximately half the sum $J'CJ$. In fact, if the diagonal and off diagonal elements of C are constants, as in simple random sampling, $J'C\boldsymbol{\mathcal{Z}}$ is exactly $0.5(J'CJ)$. One half Equation 3.12 will be used in place of $J'C\boldsymbol{\mathcal{Z}}$ in Equation 3.11.

At this point, the only information lacking from the unobserved units which prevents computation of $\tilde{V}(t_p)$ is the order of unit labels which correctly sort the observations in $\boldsymbol{Y_u}$ and $\boldsymbol{Y_v}$. How does ordering of responses facilitate computation of $\tilde{V}(t_p)$? Let us assume that the correct orderings of $\boldsymbol{Y_u}$ and $\boldsymbol{Y_v}$ are known and that o_u and o_v are the vectors which order them. That is, $\boldsymbol{Y_u}[o_u]$ and $\boldsymbol{Y_v}[o_v]$ are the correctly (ascending) ordered responses from all units in both universes. Now let $\boldsymbol{Z_{uv}}^o = I(\boldsymbol{Y_u}[o_u] < \boldsymbol{Y_v}[o_v]) + 0.5I(\boldsymbol{Y_u}[o_u] = \boldsymbol{Y_v}[o_v])$, $\boldsymbol{\mathcal{Z}}^o = vec(\boldsymbol{Z_{uv}}^o)$, $\boldsymbol{\pi_u}^o = \boldsymbol{\pi_u}[o_u]$, $\boldsymbol{\pi_v}^o = \boldsymbol{\pi_v}[o_v]$, $\boldsymbol{\pi_{uu^*}}^o = \boldsymbol{\pi_{uu^*}}[o_u,o_u]$, and $\boldsymbol{\pi_{vv^*}}^o = \boldsymbol{\pi_{vv^*}}[o_v,o_v]$. Note that $\boldsymbol{Z_{uv}}^o$ (and consequently $\boldsymbol{\mathcal{Z}}^o$) are known *under the null hypothesis* because $\boldsymbol{Y_u}[o_u] = \boldsymbol{Y_v}[o_v]$. Under the null hypothesis, $\boldsymbol{Z_{uv}}^o = \begin{bmatrix} 0.5 & 1 & 1 & \dots & 1 \\ 0 & 0.5 & 1 & & 1 \\ 0 & 0 & 0.5 & & 1 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0.5 \end{bmatrix}$. We now have all the pieces to compute $\tilde{V}(t_p)$ if ordering is known because C is reordered by computing

$$C^o = (\boldsymbol{\pi_{uu^*}}^o/\boldsymbol{\pi_u}^o\boldsymbol{\pi_u}^{o\prime}) \otimes (\boldsymbol{\pi_{vv^*}}^o/\boldsymbol{\pi_v}^o\boldsymbol{\pi_v}^{o\prime})$$

and

$$\begin{aligned}\tilde{V}(t_p) &= \left(\frac{1}{NM}\right)^2 \left[\boldsymbol{\mathcal{Z}}^{o\prime} C^o \boldsymbol{\mathcal{Z}}^o - J' C^o \boldsymbol{\mathcal{Z}}^o + \frac{1}{4} J' C^o J\right] \\ &\approx \left(\frac{1}{NM}\right)^2 \left[\boldsymbol{\mathcal{Z}}^{o\prime} C^o \boldsymbol{\mathcal{Z}}^o - \frac{1}{4} J' C J\right] \end{aligned} \tag{3.13}$$

The correct ordering of all responses might be known, for example, when an auxiliary variable is known to have perfect rank correlation with the response of interest (i.e., order of auxiliary variable is same as order of responses). In general, the correct ordering of responses is not known. However, under special designs, Equation 3.13 can be simplified. Under all designs which have constant first and second order inclusion probabilities, like simple random sampling, the inclusion probability matrices can be written as

$$\begin{aligned}\boldsymbol{\pi_u} &= \frac{1}{\bar{w}_u} J_N, \qquad \boldsymbol{\pi_v} = \frac{1}{\bar{w}_v} J_N \\ \boldsymbol{\pi_{uu^*}} &= \frac{\bar{w}_{uu^*}}{\bar{w}_u^2} J_N J_N{}' + \frac{\bar{w}_u - \bar{w}_{uu^*}}{\bar{w}_u^2} I_N \\ \boldsymbol{\pi_{vv^*}} &= \frac{\bar{w}_{vv^*}}{\bar{w}_v^2} J_N J_N{}' + \frac{\bar{w}_v - \bar{w}_{vv^*}}{\bar{w}_v^2} I_N.\end{aligned}$$

Because the inclusion probabilities can be written as scaler numbers times matrices, the scalers can be factored out the Kroneker expression of C. Also, all probabilities are constant so the ordering of C does not matter (i.e., C^o=C). Carrying out the Kroneker multiplication involved in C, $\boldsymbol{\mathcal{Z}}' C \boldsymbol{\mathcal{Z}}$ can be written as,

$$\begin{aligned}\boldsymbol{\mathcal{Z}}' C \boldsymbol{\mathcal{Z}} &= (\bar{w}_u \bar{w}_v)\, \boldsymbol{\mathcal{Z}}' (I \otimes I)\, \boldsymbol{\mathcal{Z}} \\ &+ (\bar{w}_u \bar{w}_{vv^*})\, \boldsymbol{\mathcal{Z}}' (I \otimes JJ' - I \otimes I)\, \boldsymbol{\mathcal{Z}} \\ &+ (\bar{w}_{uu^*} \bar{w}_v)\, \boldsymbol{\mathcal{Z}}' (JJ' \otimes I - I \otimes I)\, \boldsymbol{\mathcal{Z}} \\ &+ (\bar{w}_{uu^*} \bar{w}_{vv^*})\, \boldsymbol{\mathcal{Z}}' (JJ' \otimes JJ' - I \otimes JJ' - JJ' \otimes I + I \otimes I)\, \boldsymbol{\mathcal{Z}},\end{aligned} \tag{3.14}$$

where all I and J matrices are of size N. This partitioning of $\boldsymbol{\mathcal{Z}}' C \boldsymbol{\mathcal{Z}}$ is reminiscent of a 2-way ANOVA partitioning of the variance of t_p.

Under the null hypothesis, the general $\boldsymbol{\mathcal{Z}}$ in Equation 3.14 is a reordered version of $\boldsymbol{\mathcal{Z}}^o$ and the various sums-of-squares in Equation 3.14 can be found by counting. Recall that under the null hypothesis and assuming no ties within either population,

$\mathcal{Z}$ contains exactly $[N(N-1)/2]$ 1's, $[N(N-1)/2]$ 0's, and $[N]$ 0.5's. Under the null hypothesis,

$$\mathcal{Z}'(I \otimes I)\mathcal{Z} = \frac{N(2N-1)}{4}$$

$$\mathcal{Z}'(I \otimes JJ' - I \otimes I)\mathcal{Z} = \frac{N(N-1)(2N-1)}{6}$$

$$\mathcal{Z}'(JJ' \otimes I - I \otimes I)\mathcal{Z} = \frac{N(N-1)(2N-1)}{6}$$

$$\mathcal{Z}'(JJ' \otimes JJ' - I \otimes JJ' - JJ' \otimes I + I \otimes I)\mathcal{Z} = \frac{3N^4 - N(2N-1)(4N-1)}{12}$$

$\tilde{V}(t_p)$ under the null hypothesis and under designs with constant first and second order inclusion probabilities can now be rewritten in a simple form by first substituting the above sums-of-squares into Equation 3.14, the resulting $\mathcal{Z}'\mathrm{C}\mathcal{Z}$ into Equation 3.13 for $\mathcal{Z}^{2\prime}\mathrm{C}\mathcal{Z}^2$, then use Equation 3.12 in place of $J'\mathrm{C}J$ and collect terms. The resulting expression for $\tilde{V}(t_p)$ is,

$$\tilde{V}(t_p) = C_1\bar{w}_u\bar{w}_v + C_2(\bar{w}_u\bar{w}_{vv^*} + \bar{w}_v\bar{w}_{uu^*}) - C_3\bar{w}_{uu^*}\bar{w}_{vv^*}, \tag{3.15}$$

where $C_1 = N(N-1)/4$, $C_2 = N(N-1)(N-2)/12$, and $C_3 = N(N-1)(2N-1)/12$.

Equation 3.15 is Equation 3.10 under the null hypothesis and under designs with constant first and second order inclusion probabilities, like simple random sampling. In fact, under simple random sampling Equation 3.15 reduces to $(nm)^{-2}V_c(R)$, where $V_c(R)$ is given in Equation 3.7. Equation 3.15 is useful in that it suggests an approximate variance for t_p under designs which are not simple random or do not have constant inclusion probabilities.

Recall that if design inclusion probabilities are not constant but the ordering of responses is known, Equation 3.13 gives the correct variance for t_p under the null hypothesis. However, if design inclusion probabilities are not constant and the correct order of responses is not known, the correct variance of t_p is one of the $N^2!$ possible $\tilde{V}(t_p)$ arising from all possible permutations of the order vectors o_u and o_v. That is, we know now many 1's, 0's, and 0.5's are contained in $\mathcal{Z}$, but because many are unobserved, we do not know which design weights in C are multiplied by 1's, which

are multiplied by 0's, and which are multiplied by 0.5's. The correct value of $\tilde{V}(t_p)$ will never be known in this situation unless a census is taken.

Under non-constant inclusion probabilities and if time permitted its computation, a reasonable value for $\tilde{V}(t_p)$ would be the average of all $N^2!$ $\tilde{V}(t_p)$'s computed from every permutation of $\mathcal{Z}$. Computing Equation 3.10 for every permutation of $\mathcal{Z}$ is unreasonable for N over 5 units (1.55e25 permutations for N=5). Luckily, the average of $\tilde{V}(t_p)$ over all $N^2!$ permutations is well approximated by Equation 3.15. The average $\tilde{V}(t_p)$ is well approximated by Equation 3.15 because averaging $\tilde{V}(t_p)$ over permutations of $\mathcal{Z}$ fundamentally averages over inclusion probabilities due to the 0's and 1's in $\mathcal{Z}$ which act to include and exclude values in C across permutations. Using Equation 3.15 to approximate the average $\tilde{V}(t_p)$ over permutations is parallel to approximating $\frac{1}{n}\sum_{x_i} f(x_i)$ with $f(\bar{x})$, where $f()$ is a quadratic function. In some regions of its domain, the quadratic $f()$ is nearly linear and the approximation is very good.

The proposed test is almost complete. To complete the proposed test a probability level must be assigned to the actual observed value of t_p. If this observed value rarely occurs by chance, there is strong evidence that the null hypothesis is not true. The true distribution of t_p is generally very complicated because of its dependence on the design and can only be worked out for a small number of special designs (namely SRS and close relatives). As in many hypothesis testing situations, a reasonable approximation of the true distribution of t_p makes the testing procedure based on t_p useful, but not perfect. This study proposes a null distribution for t_p based on empirical studies of the behavior of t_p under the null hypothesis and different designs. In the empirical studies reported below, the Beta distribution with parameters estimated via method-of-moments (MOM) provided a reasonable approximation to the distribution of t_p under the null hypothesis.

In order to use the Beta distribution to assign probability levels to t_p, its parameters must be estimated. Mood et al. (1974) give the mean and variance of a Beta(a,b) distribution defined on the interval (0,1) as $a/(a+b)$ and $ab/(a+b+1)(a+b)^2$. Setting these two equations equal to 0.5 ($=\mu_z$under the null hypothesis) and $\tilde{V}(t_p)$ and

solving for a and b yields the MOM estimators,

$$\hat{a} = \hat{b} = \frac{1}{8\tilde{V}(t_p)} - \frac{1}{2}$$

A problem arises when $\hat{a}$ and $\hat{b}$ are close to or less than one. When $\hat{a}$ and $\hat{b}$ are close to or less than one, the resulting Beta density is either "flat" or "U" shaped and in general provides a very poor approximation to the distribution of t_p, which is generally unimodal. An *ad hoc* rule is established to deal with this situation. If $\hat{a}$ and $\hat{b}$ are less than 1.25, a Normal distribution with mean 0.5 and variance $\tilde{V}(t_p)$ is used to approximate the distribution of t_p.

In summary, the probability level for a two-sided proposed test is determined as follows: (one-sided significance levels are half the two-sided because the particular Beta used is symmetric around 0.5): If $\hat{a}$ and $\hat{b}$ are *greater* than 1.25, the area to the left of the observed t_p under a Beta($\hat{a}$,$\hat{b}$) distribution is calculated, call this area p^*; If $\hat{a}$ and $\hat{b}$ are *less* than 1.25, the area to the left of the observed t_p under a Normal(0.5, $\tilde{V}(t_p)$) distribution is calculated, again, call this area p^*; If $p^* < 0.5$, the two-sided significance level of $t_p$is $2p^*$; If $p^* > 0.5$, the two-sided significance of t_p is $2(1 - p^*)$. The null hypothesis is typically rejected if the significance level of t_p is less than some number α, where α is the nominal level of the test.

3.4 Methods: Simulation Experiments

This section describes computer simulations designed to assess the size (percent rejections of a true null hypothesis) and power (percent rejections of a false null hypothesis) of the test procedures. The populations sampled and simulation parameters are described. The outline (or algorithm) for the simulation is given. A description of grouped GRS samples and how they are taken appears in Appendix B.

Responses used in the simulations were from three sources: (1) acid neutralizing capacity (ANC) of 110 lakes in the northeastern United States collected as part of the National Lake Survey (Linthurst et al., 1986), (2) number of rental dwelling on 270 blocks in Ward 1 of Fall River, Massachusetts, collected as part of the 1950

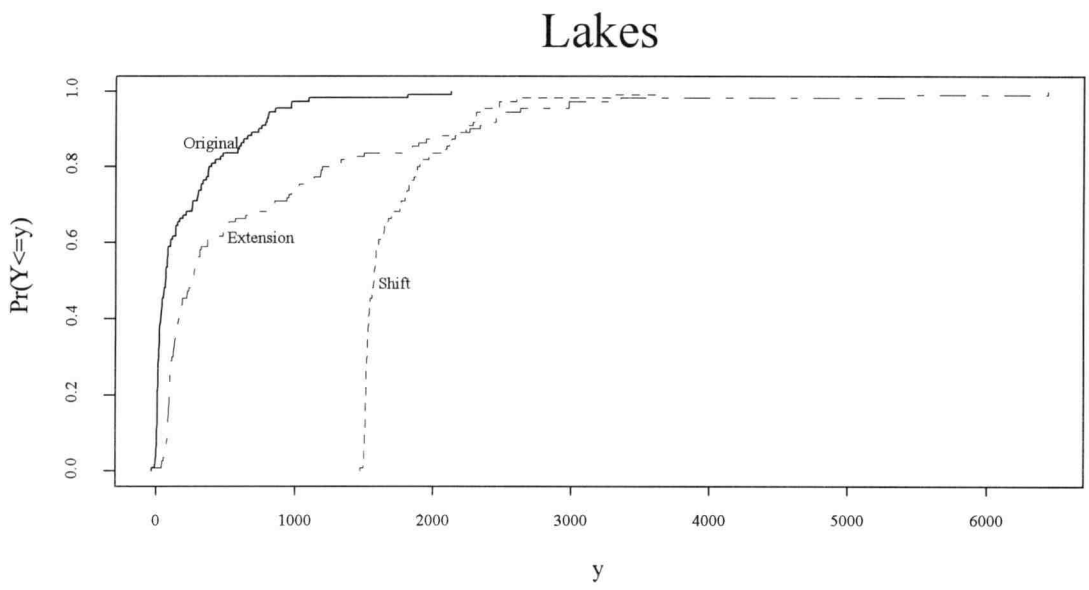

Figure 3.1: Lakes population CDF (solid) and examples of shift and extension CDF's (dashed) used in the simulation.

Census and appearing in Kish (1965, Appendix E), and (3) 250 randomly generated deviates from a Normal distribution having mean 100 and standard deviation 25. For the simulations, these data comprised the response vector $\boldsymbol{Y_u}$. The three population CDF's ($F_u(t)$) and examples of "shift" and "extension" alternatives appear in Figure 3.1 through Figure 3.3. The "shift" and "extension" alternatives are described in the outline of the simulation below.

If first order inclusion probabilities are not constant, there exists a correlation between inclusion probabilities and the unknown responses. When estimation is a primary goal, researchers usually hope for high correlation between inclusion probabilities and responses because the usual Horvitz-Thompson estimator has low variance in this case (Cassel et al., 1977; Särndal et al., 1992). However, researchers rarely if ever know the true correlation between responses and inclusion probabilities. It would therefore be desirable to use an (unbiased) estimator which has low variance regardless of the correlation between inclusion probabilities and responses. When testing, the same desire applies. It would be desirable to use a test whose size and power is

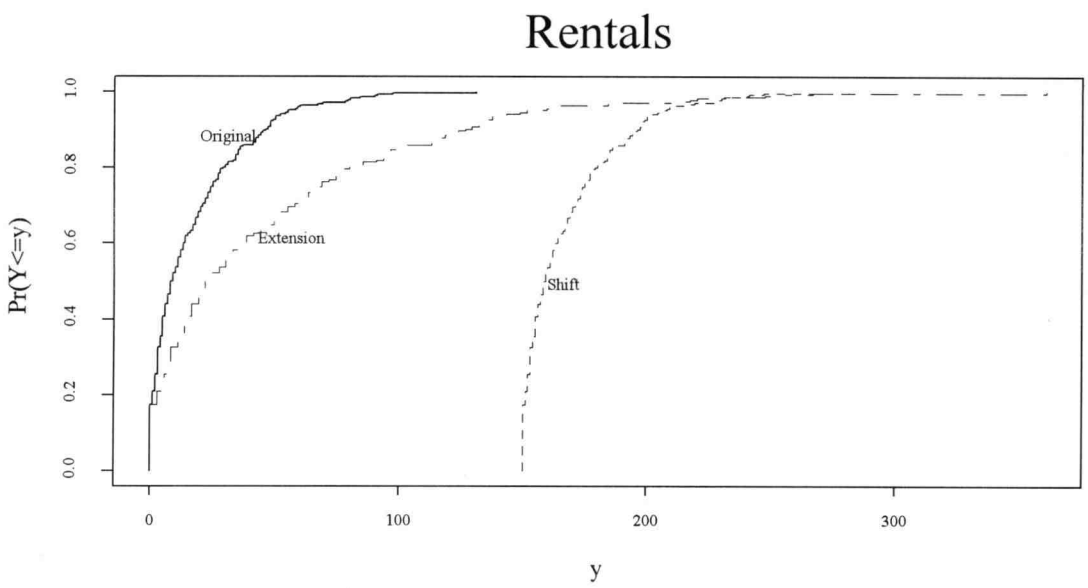

Figure 3.2: Rentals population CDF (solid) and examples of shift and extension CDF's (dashed) used in the simulation.

relatively insensitive to changes in the correlation between inclusion probabilities and responses, mainly because this correlation is generally unknown.

One design parameter which the computer simulations vary is the rank correlation between inclusion probabilities and responses. When ranks are used, as in this paper, correlation between inclusion probabilities and responses is equivalent to the extent to which inclusion probabilities sort responses. A useful quantification of this sorting is Spearman's rank correlation coefficient (Zar, 1974). When the rank correlation between two lists is 1, each list perfectly sorts the other in the same order. When rank correlation is -1, each list perfectly sorts the other in the opposite direction. Recall that if the rank correlation between responses and some list of numbers is 1, Equation 3.13 is the (1st-order) variance of t_p.

Another parameter varied in the simulations is the variance of inclusion probabilities. Variance of inclusion probabilities was chosen as a parameter in the simulation because it measures the "closeness" of the design to simple random sampling. Some other measure of dispersion in inclusion probabilities could have been chosen. If the variance of inclusion probabilities is high, there are units in the population which

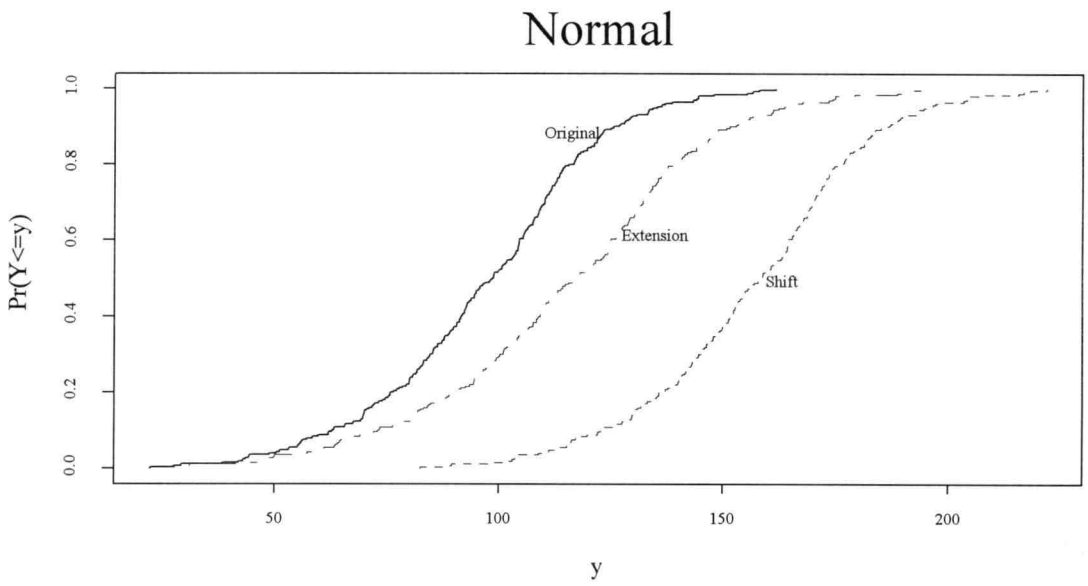

Figure 3.3: Normal population CDF (solid) and examples of shift and extension CDF's (dashed) used in the simulation.

are sampled much more frequently than other units. Due to the constraint that all inclusion probabilities lie between zero and one and sum to n, the maximum variance of first order inclusion probabilities, denoted $\sigma_{\pi_{max}}$, is, $\sigma_{\pi_{max}} = \frac{n(N-n)}{N(N-1)}$. This variance results from a degenerate design which samples n units with probability 1. Although this degenerate design is not interesting, it provides a basis for judging the magnitude of inclusion probability variance of other designs. Variance in inclusion probabilities for designs in the simulation will be expressed as a percentage of this number (i.e., 0% corresponds to simple random sampling, 100% corresponds with the completely degenerate design just mentioned).

A full list of simulation parameters, along with their labels, is:

- **"Lakes", "Rentals", or "Normal"**: The population data loaded into $\boldsymbol{Y_u}$.
- **n and m**: Sample sizes.
- $\sigma^2_{\pi_u}$ **and** $\sigma^2_{\pi_v}$: Variance of the inclusion probabilities from both populations.
- g_u **and** g_v: The number of groups in the grouped GRS design from each population.

- ρ_u **and** ρ_v: Rank correlation between responses and inclusion probabilities.
- k: The amount of shift or extension in $\boldsymbol{Y_v}$ if the null hypothesis is not true.

Under a given set of parameters, the simulation proceeded as follows:

1. Let y_{min} be the smallest element in $\boldsymbol{Y_u}$, and set

$$\boldsymbol{Y_v} = \begin{cases} \boldsymbol{Y_u} & \text{if the null hypothesis is true} \\ \boldsymbol{Y_u} + k & \text{under } \textit{shift} \text{ alternative} \\ k(\boldsymbol{Y_u} - y_{min}) + y_{min} & \text{under } \textit{extension} \text{ alternative} \end{cases}$$

2. Construct $\boldsymbol{\pi_u}$ so that $\boldsymbol{\pi_u}$ contains only g_u distinct values, the variance of elements in $\boldsymbol{\pi_u}$ is $\sigma^2_{\pi_u}$, and the rank correlation between $\boldsymbol{\pi_u}$ and $\boldsymbol{Y_u}$ is ρ_u.
3. Repeat the above for $\boldsymbol{\pi_v}$.
4. Repeat the following 1000 times:
   (a) Draw a size n GRS sample from $\boldsymbol{Y_u}$ using $\boldsymbol{\pi_u}$.
   (b) Draw a size m GRS sample from $\boldsymbol{Y_v}$ using $\boldsymbol{\pi_v}$.
   (c) Apply the naive, corrected, and proposed tests, using Overton's approximation to second order inclusion probabilities when necessary.
   (d) For each test, record whether or not the null hypothesis is rejected.

3.5 Results

This section presents results of the simulation experiments. Results presented in this section show the influence of each set of simulation parameters on test size. The results of all simulations were remarkable similar. The graphs and simulations chosen for presentation here attempt to isolate the influence of certain parameters by fixing the remaining ones. The parameters which are isolated are the ones which, in general,

had the most influence on test size. The remaining simulation parameters had less influence on the tests and therefore the graphs presented here were, in general, the same general shape across values of the other parameters. Space considerations did not warrant separate presentation of all graphs. The simulation parameters which are not specifically highlighted in these results because they had relatively little influence were the population (Lakes, Rentals, and Normal), sample size (n and m), and number of design groups (g_u and g_v).

Power of the tests is a secondary consideration and is only presented for a small number of simulations because generally the tests differed substantially in size. Power comparisons make little sense in cases where size differs.

The naive and corrected Wilcoxon tests, which differ only in a multiplicative correction to variance, performed similarly in all simulations in that the corrected Wilcoxon always dominated the naive Wilcoxon. This result is not surprising because the correction reduces estimated variance. Because of the dominance of the corrected Wilcoxon over the naive Wilcoxon, results will focus on comparing the corrected Wilcoxon to the proposed test.

This section starts by presenting the true and approximate test statistic distributions in three representative cases.

To illustrate what is going on "behind the scenes" in the simulations, the actual and approximating distributions of t_p and R under the null hypothesis are presented from three representative simulations. Figure 3.4 presents (Gaussian kernel) smoothed density estimates in three simulations which primarily represent (a) simple random sampling ($\sigma^2_{\pi_u}=\sigma^2_{\pi_v}=0$), (b) GRS sampling with ρ_u= -0.5, and ρ_v=0.5, and (c) GRS sampling with ρ_u= 0.5 and ρ_v=0. The panels of Figure 3.4 differ in that (a) panels sampled from the Rentals population, (b) panels sampled from the Normal population with $\sigma^2_{\pi_u}=\sigma^2_{\pi_v}=25\%$ of maximum, and (c) panels sampled from the Normal population with $\sigma^2_{\pi_u}=\sigma^2_{\pi_v}=30\%$ of maximum. Sample sizes in all panels were n=m=27. Solid lines in Figure 3.4 represent actual density functions of the statistics while dashed lines represent the (Beta or Normal) nominal distributions used to assign significance levels. Size of the test procedures is the area under the solid line

to the left of c_l and to the right of c_u, where c_l is the $\alpha/2$ quantile from the nominal (dashed) distribution and c_u is the $1 - \alpha/2$ quantile from the nominal distribution.

The simulations presented in Figure 3.4 were chosen to illustrate that under some conditions the Wilcoxon rank sum performs as predicted but that under other conditions the actual distribution of R is shifted to the right or left of its nominal distribution. Figure 3.4 also illustrates that under the same conditions, the distribution of t_p remains relatively unchanged. The general phenomenon of relative stability in the distribution of t_p and relative instability in the distribution of R was observed throughout the simulations.

The influences of design correlations (ρ_u and ρ_v) and inclusion probability variance ($\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$) on test size are treated in separate subsections below. The other simulation parameters, by and large, had less influence on test size than correlations and inclusion probability variance or affected both tests equally and predictably. The performance (size and power) of both tests improved with increased sample sizes, but did not seem to be related to the fraction of the population being sampled (see below). The number of design groups (g_u and g_v) had a moderate influence on test size. Simple random sampling forced g_u or g_v= 1. If g_u or g_v equaled two or three, there were relatively few samples with members from differing groups and, in some cases, caused the true distribution of t_p or R to be highly skewed or even bimodal. When the number of design groups was increased and everything else remained the same, the probability of obtaining samples with units from different groups was increased and skewness in the distribution of t_p was reduced because the Horvitz-Thompson weights used in t_p were less dispersed (i.e., closer together). However, the influence that the number of groups had on test size was almost entirely dependent upon other factors like inclusion probability variance and design correlation. If, for example, inclusion probability variance was small, changes in the number of groups had little influence. If inclusion probability variance was high, changes in the number of design groups had relatively more influence.

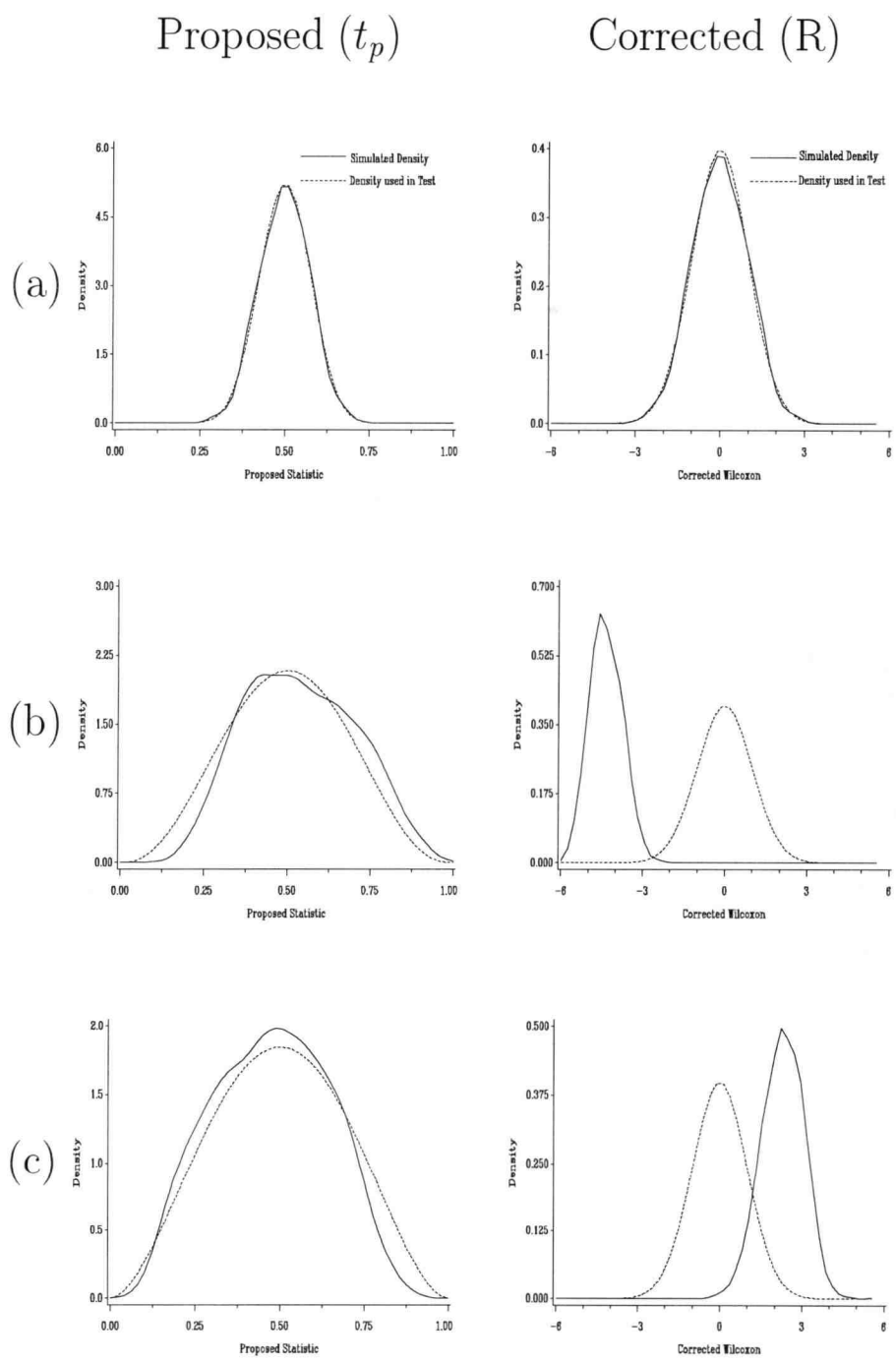

Figure 3.4: Actual and nominal distributions of t_p and R under simple random sampling (panels (a)) and GRS sampling under differing rank correlations (panels (b) and (c)). In (b), ρ_u= -0.5, ρ_v=0.5. In (c), ρ_u=0.5, ρ_v=0. See text for values of remaining simulation parameters.

3.5.1 Influence of Design Correlations

Design correlations (ρ_u and ρ_v) had a tremendous effect on test size in the simulations. This section focuses on the effects of varying ρ_u and ρ_v by keeping the remaining simulation parameters mostly constant.

Simulations showing the effects of changing ρ_u and $\rho_v$are presented in Figure 3.5 through Figure 3.8. Each figure presents a view or multiple views of three dimensional surfaces which have ρ_u and ρ_v as their horizontal coordinates and test size as their vertical coordinate. Values of the simulation parameters used to produce each figure are given in the captions. Test size (vertical dimension) in each figure was calculated as proportion of 1000 iterations on which a true null hypothesis was rejected. Ideally, the size of both tests would be 5% for all ρ_u and ρ_v and both surfaces would plot as flat surface at a vertical coordinate 0.05. In each figure, size of the proposed test is the lower surface (little cubes) while size of the corrected Wilcoxon is the upper surface with the "trough" (little circles). Based on the variance of rejections over all simulations and assuming rejections follow a binomial distribution, the width of an approximate 95% confidence interval for true test size is ± 0.03 for points around 0.5 and ± 0.014 for points in the neighborhood of 0.05. The minimum, mean, and maximum vertical coordinate from each figure is presented in Table 3.1 because exact vertical coordinates in the graphs are difficult to read.

In general, the surfaces in Figure 3.5 through Figure 3.8 have similar shape. The corrected Wilcoxon test displayed extreme violations in size when the design correlations were not equal. In fact, the corrected Wilcoxon always rejected the true null hypothesis (i.e., Type I error rate = 100%) in many cases when $\rho_u \neq \rho_v$ and the absolute value of each was large. The liberalness of the corrected Wilcoxon test when $\rho_u \neq \rho_v$ is displayed in the figures as a deep "trough" along the line where $\rho_u = \rho_v$. Figure 3.8 is presented for comparison to Figure 3.5 in order to show that orientation of the "trough" changed as other simulation parameters changed. The "trough" in Figure 3.8 is oriented along a different line in the (ρ_u, ρ_v) plane in part because $\boldsymbol{\pi_u}$ was small and changes in it effected test size less than changes in $\boldsymbol{\pi_v}$. This change

in orientation of the "trough" in Figure 3.8 when compared to Figure 3.5 indicates an interactive effect of inclusion probabilities variance and design correlations on test size.

In general, simulation error was small enough that none of the surfaces in Figure 3.5 through Figure 3.8 could be regarded as perfectly flat. However, surfaces for the proposed test were typically much flatter and within simulation error of 5% over a much larger proportion of the (ρ_u,ρ_v) plane than those for the corrected Wilcoxon. The size of both tests increased as ρ_u and ρ_v grew large in opposite directions. However, even in extreme cases when $\rho_u = 1$ and $\rho_v = -1$, (i.e., populations sorted in opposite directions) size of the proposed test was typically around 5% to 6% too large (Table 3.1) while the size of the Wilcoxon procedure was typically 95% too large.

Figure 3.6 is presented to show the effects of lowering design inclusion probability variance. The only difference between panel (a) and panel (b) of Figure 3.6 is that inclusion probability variance changes from 25% in (a) to 5% in (b). Both tests performed slightly better in the simulations of panel (b) than they did in the simulations of panel (a). The surface for the proposed test which was relatively flat in panel (a) flattened out even farther in panel (b) (Table 3.1), while the "trough" in the surface for the Wilcoxon statistic is visibly wider in panel (b).

By comparing Figure 3.5 though Figure 3.8, one can see that the influence of changes in ρ_u and ρ_v on test size was relatively unrelated to the shape of the population or the fraction of the population sampled. Fraction of population sampled changed from 11% from the Normal population in Figure 3.5 and Figure 3.8 to 10% from the Rentals population in Figure 3.6 to 25% from the Lakes population in Figure 3.7. This result may not be surprising given the rank based approach of the tests. This result was verified in other simulations not presented here. In all simulations, test performance improved as sample size increased and performance did not seem to be related to the fraction of the population being sampled.

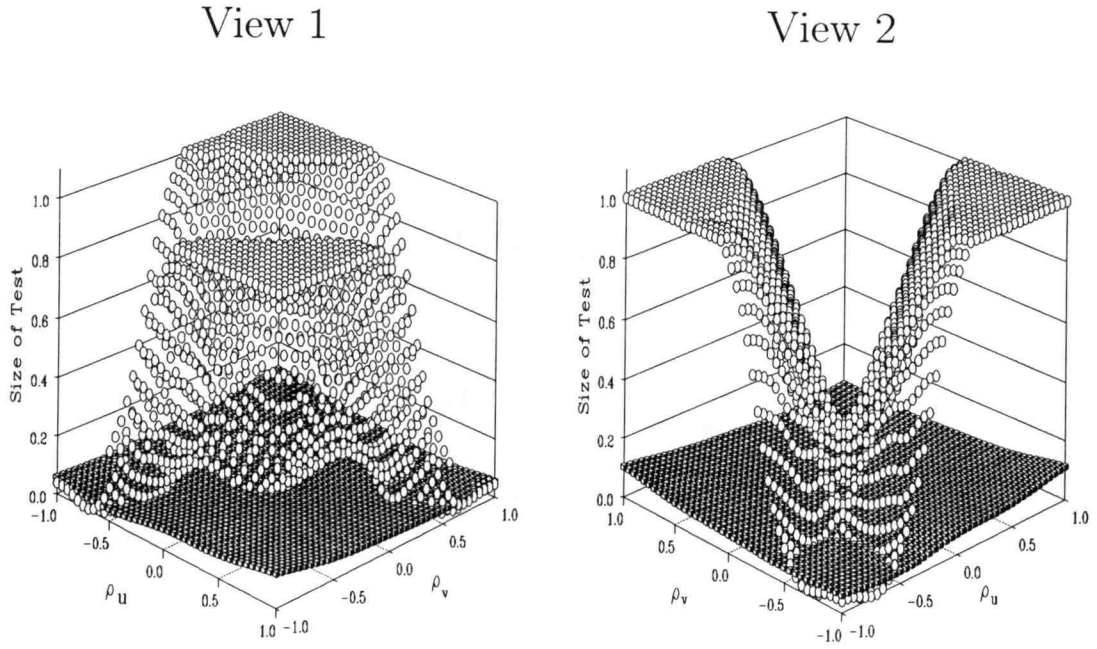

Figure 3.5: Size of proposed test and corrected Wilcoxon under varying ρ_u and ρ_v. Two views of same surface. Sampling from Normal populations with g_u=g_v=5, n=m=27, and $\sigma^2_{\pi_u}$=$\sigma^2_{\pi_v}$=25% of maximum.

	Proposed Test (t_p)			Corrected Wilcoxon (R)		
Figure	Mean	Min	Max	Mean	Min	Max
3.5	0.0519	0.020	0.110	0.5907	0.020	1.000
3.6 (a)	0.0513	0.017	0.113	0.5898	0.014	1.000
3.6 (b)	0.0671	0.059	0.084	0.4580	0.035	0.998
3.7	0.0472	0.028	0.089	0.5690	0.018	1.000
3.8	0.0630	0.040	0.095	0.6022	0.029	1.000
3.9 (a)	0.0281	0.000	0.065	0.2802	0.011	0.915
3.9 (b)	0.0277	0.000	0.055	0.4836	0.020	0.910
3.10	0.0485	0.023	0.068	0.2917	0.012	0.932
3.11	0.0248	0.000	0.066	0.2711	0.047	0.565

Table 3.1: Average, minimum, and maximum test size for simulations in Figure 3.5 through Figure 3.11.

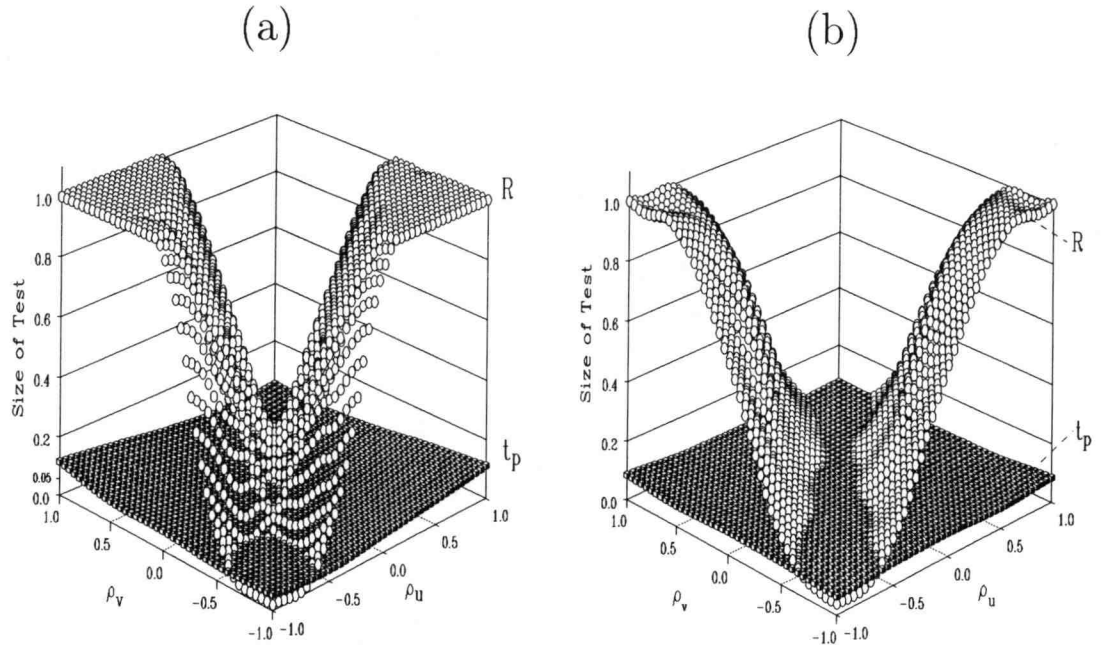

Figure 3.6: Size of proposed test (lower surface, cubes) and corrected Wilcoxon (upper surface, circles) under varying levels of coru and ρ_v. Sampling from Rentals population with g_u=g_v=5 and n=m=27. Panel (a), $\sigma^2_{\pi_u}$=$\sigma^2_{\pi_v}$=25% of maximum. Panel (b), $\sigma^2_{\pi_u}$=$\sigma^2_{\pi_v}$=5% of maximum.

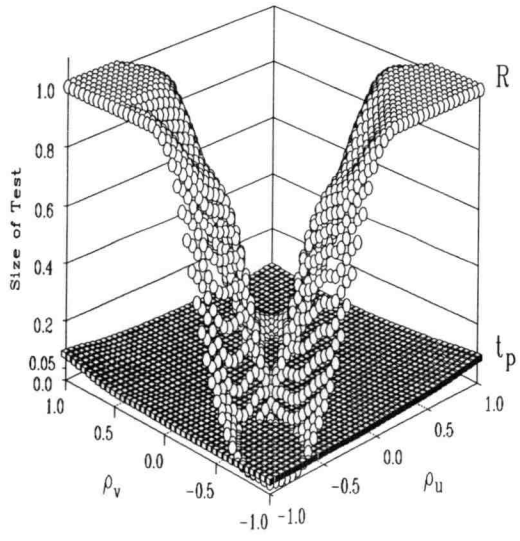

Figure 3.7: Size of proposed test and corrected Wilcoxon under varying ρ_u and ρ_v. Sampling from Lakes population, g_u=g_v=55, n=m=27, and $\sigma^2_{\pi_u}$=$\sigma^2_{\pi_v}$=25% of maximum.

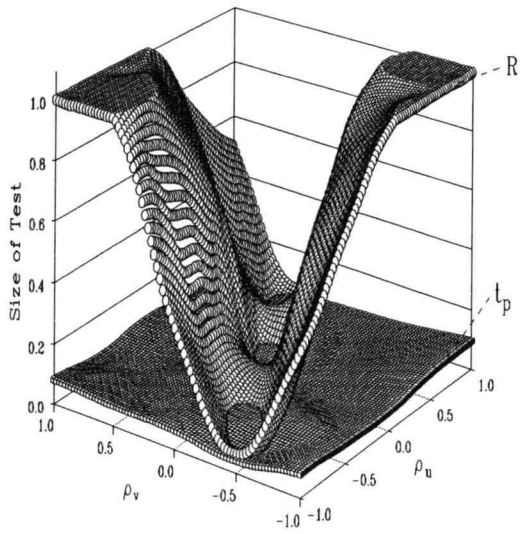

Figure 3.8: Size of proposed test and corrected Wilcoxon under varying ρ_u and ρ_v. Sampling from Normal population with g_u=g_v=5, n=m=27, $\sigma^2_{\pi_u}$=5%, and $\sigma^2_{\pi_v}$=20% of maximum.

3.5.2 Influence of Inclusion Probability Variance

In consort with changes in design correlation, changes in the variance of inclusion probabilities had a tremendous influence on test size. This section attempts to isolate and characterize some of these effects.

Figure 3.9 through Figure 3.11 present size of the proposed and corrected procedures as the variance of $\boldsymbol{\pi_u}$ and $\boldsymbol{\pi_v}$ changes. These figures are parallel to the figures in Section 3.5.1 in that they contain three dimensional surfaces with test size as the vertical dimension and $\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$ as the two horizontal coordinates. Values of the other simulation parameters can be extracted from the caption of each figure. Again, the surface for size of the proposed test is made up of little cubes and is usually flatter and below the surface for size of the corrected Wilcoxon which is made up of little circles. The mean, minimum, and maximum vertical coordinates for each surface appear in Table 3.1. When $\sigma^2_{\pi_u}$ or $\sigma^2_{\pi_v}$ was exactly zero, simple random sampling was performed from that population and correlation between inclusion probabilities

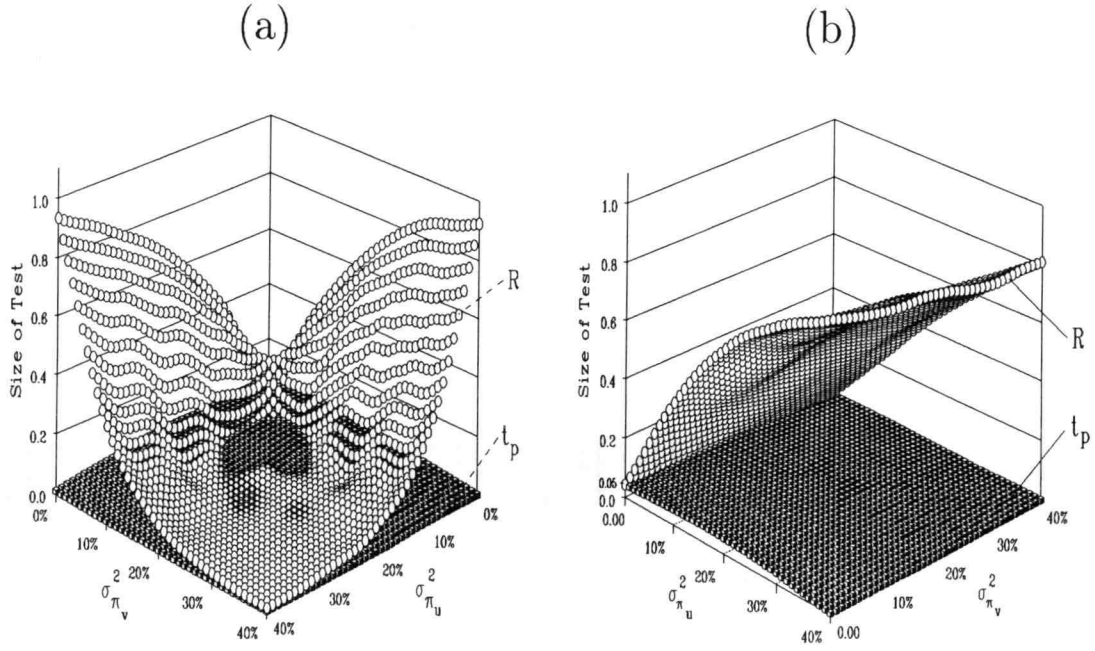

Figure 3.9: Test size under varying $\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$. Sampling from Normal population with g_u=g_v=5 and n=m=27. Panel (a), ρ_u=ρ_v=0.5. Panel (b), ρ_u=0.5 ρ_v=0.

and responses was undefined (not computed). As $\sigma^2_{\pi_u}$ or $\sigma^2_{\pi_v}$ grew large, an increasing number of units were sampled with probability one and at the same time an increasing number of units were sampled with probability very close to zero.

Figure 3.9 through Figure 3.11 show a fundamental difference in behavior of the proposed and corrected tests when inclusion probability variance increases. Under simple random sampling ($\sigma^2_{\pi_u}$=$\sigma^2_{\pi_v}$=0), both tests have size near 5%. This can be seen clearly in Figure 3.10, view 2. However, as the sampling design became less like simple random sampling ($\sigma^2_{\pi_u}$,$\sigma^2_{\pi_v}$>0), the proposed test typically became conservative (size < 5%) while the corrected Wilcoxon procedure became extremely liberal (size >> 5%) in some cases (Table 3.1). Both tests typically lost all practical value when inclusion probability variance climbed above 40% to 50% of maximum because the tests either never rejected (Type II error 100%) or the Wilcoxon always rejected (Type I error 100%). Designs with inclusion probability variance above 40% sampled many units with probability one and many units with (essentially) probability zero and it is

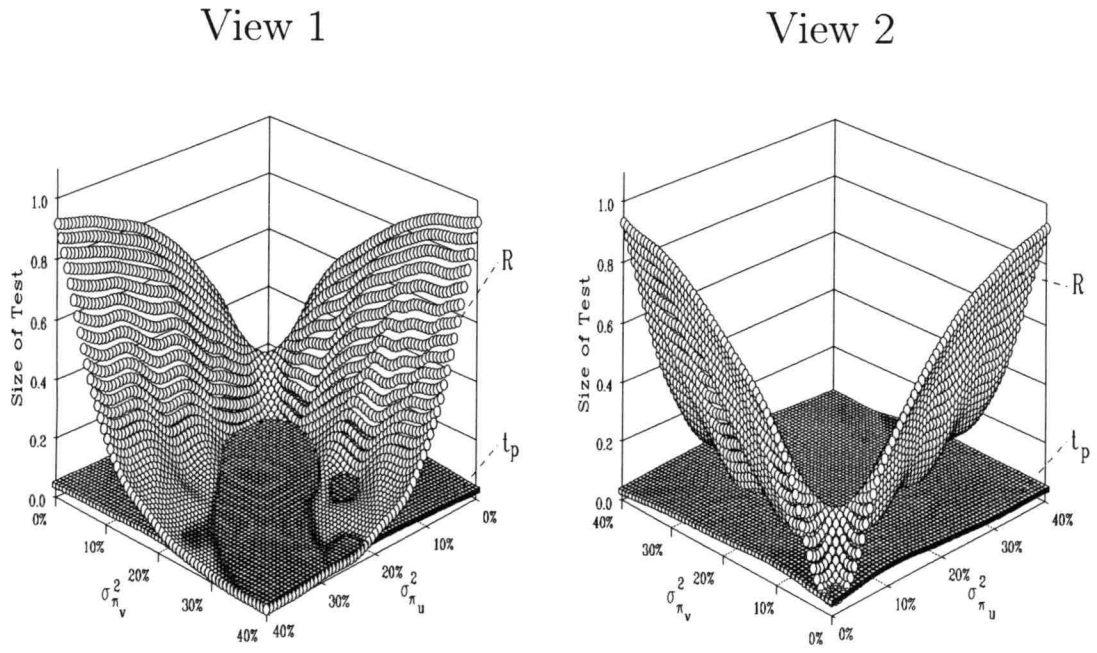

Figure 3.10: Size of proposed test and corrected Wilcoxon under varying $\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$. Two views of same surface. Sampling from Rentals populations with g_u=g_v=54, n=m=50, and ρ_u=ρ_v=0.5.

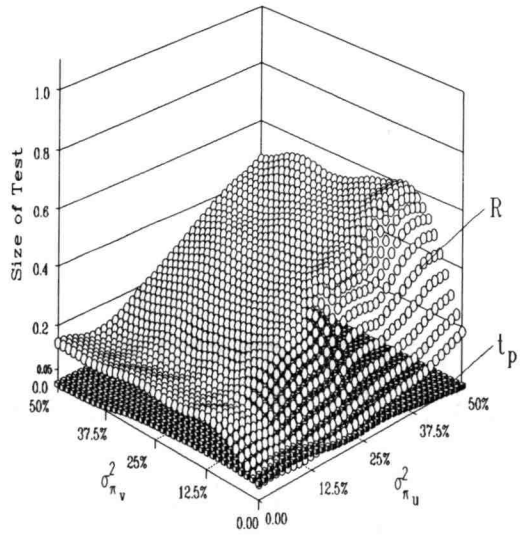

Figure 3.11: Test size under varying $\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$. Sampling from Lakes population with g_u=g_v=5, n=m=27, ρ_u=0, and ρ_v= 0.5.

unlikely that any test will perform well in these situations. Again, the proposed test remained within simulation error of 5% over a much larger set of $(\sigma^2_{\pi_u},\sigma^2_{\pi_v})$ than did the corrected Wilcoxon and had the added benefit of always becoming conservative rather than liberal when inclusion probability variance became large.

Comparing panels (a) and (b) in Figure 3.9 again shows the interactive effects of changes in $\sigma^2_{\pi_u}$, $\sigma^2_{\pi_v}$, ρ_u, and ρ_v on size of the Wilcoxon test. The only difference between panel (a) and panel (b) in Figure 3.9 is that ρ_v changed from 0.5 in (a) to 0.0 in (b) and yet the Wilcoxon surfaces look completely different. The Wilcoxon surface in (b) is relatively flat in one direction and relatively steep in the other direction. The dimension in which test size changes most rapidly is the dimension corresponding to the population with higher correlation between inclusion probabilities and responses. This illustrates that size of the Wilcoxon test is much more sensitive to changes in inclusion probability variance when design correlation with responses is high. Size of the proposed test did not display increased sensitivity to changes in inclusion probability variance when design correlation is changed.

Panels (a) and (b) of Figure 3.9, and Figure 3.10, illustrate another point regarding the corrected Wilcoxon test. If design correlations are equal in both populations (i.e., populations sorted in same order) then eventually as $\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$ increase the same units are being sampled from both populations with high probability and the populations appear similar to the Wilcoxon test. If, on the other hand, design correlations are not equal across populations, then eventually as $\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$ increase different units are entering the two samples with high probability and the populations appear dissimilar to the Wilcoxon test. These statements are reflected in the fact that the Wilcoxon surface in panel (a) of Figure 3.9 goes to zero as both $\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$ increase while in panel (b) size of the Wilcoxon goes to one as $\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$ increase.

Figure 3.11 shows that size of the Wilcoxon test is mildly sensitive to shape of the population CDF being sampled. Figure 3.11 should be compared to panel (b) of Figure 3.9 where the only difference is the population being sampled. Note that Figure 3.11 and panel (b) of Figure 3.9 have different viewpoints. Again, the shape

of surfaces for the proposed test appeared similar across populations (compare (b) of Figure 3.9 to Figure 3.11).

3.5.3 Power

Power comparisons among the tests were of less utility than anticipated because the proposed, naive, and corrected tests generally have very different size. Power was considered mostly to verify that the proposed test displayed increasing power to detect an ordered set of certain alternatives. This section presents power of all three tests in about the only situation where the Wilcoxon tests consistently had correct size.

Size of the corrected Wilcoxon test was consistently at or below the nominal level when the same design correlations were used in both populations and inclusion probability variance was the same in each population. By setting $\rho_u=\rho_v$ and varying the common value of $\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$, the design remained in the "trough" displayed in Figure 3.5 through Figure 3.8 where the corrected Wilcoxon had correct or conservative size. Figure 3.12 contains four plots of the power of each test to detect a shift alternative. The upper left panel with $\sigma^2_{\pi_u}=\sigma^2_{\pi_v}=0$ corresponds to simple random sampling; the other three panels depict designs progressively violating simple random sampling.

As expected, the proposed and corrected Wilcoxon tests were virtually identical under simple random sampling. Recall that t_p is a linear combination of R under simple random sampling. As inclusion probability variance increased, the size of all three tests decreased slightly and power of the corrected Wilcoxon dropped below that of the proposed test. Power of the naive test was always below that of the corrected test. The power of all three tests to detect the same size shift decreased as inclusion probability variance increased. Power of the proposed test increased as the amount of shift increased.

Figure 3.13 presents typical power curves for detecting an extension alternative. Figure 3.13 is comparable to Figure 3.12 in that design correlations are held constant

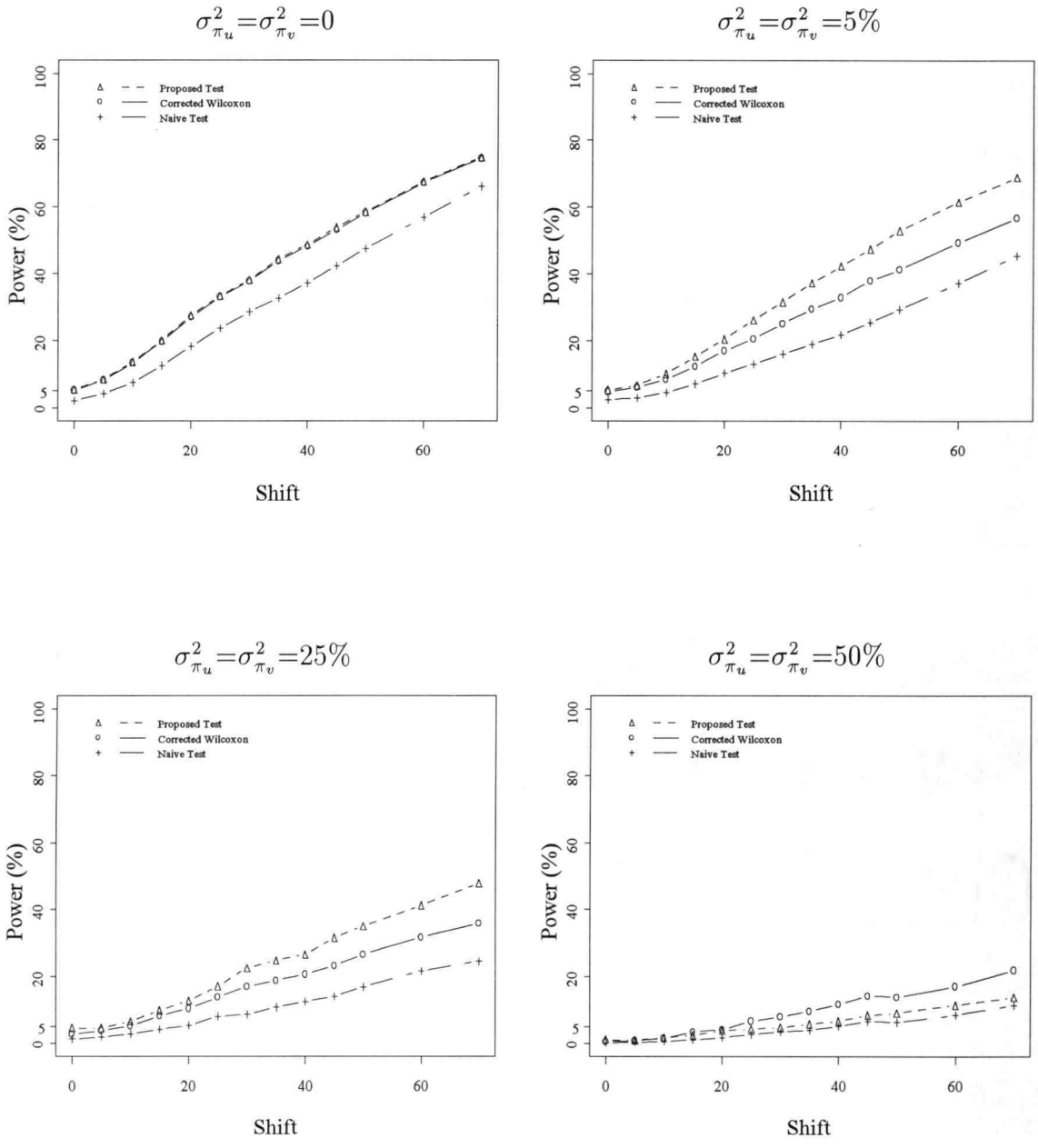

Figure 3.12: Power of the tests under *shift* alternatives. Sampling from the Lakes population with n=m=27, g_u=g_v=103, and ρ_u=ρ_v.

and inclusion probability variance is changed from panel to panel. Again, power of the proposed test was identical to that of the corrected Wilcoxon under simple random sampling. However, Figure 3.13 shows some interesting behavior of the tests under designs with non-zero inclusion probability variance. Power of the Wilcoxon statistic was actually higher than that of the proposed test in the non-simple random sample

simulations. Note that ρ_u and ρ_v are high in Figure 3.13 and that as $\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$ increase a relatively few number of distinct units are actually being sampled. For example, with $\sigma^2_{\pi_u}$ and $\sigma^2_{\pi_v}$ high enough, there may be 20 units out of the sample of size 27 units which are sampled with probability one. If those units which are always sampled change slightly, the change will appear real and consistent to the Wilcoxon procedure which does not take into account sample weights. On the other hand, the proposed test down weights the units with are always sampled and so the change in these units appears less large. Figure 3.13 was presented to point out this behavior in the Wilcoxon statistic and this behavior might, in some senses, be called a flaw in the Wilcoxon procedure even though power of the Wilcoxon was high. The fact that the proposed test had lower power in these relatively extreme (large ρ_u, ρ_v, $\sigma^2_{\pi_u}$, and $\sigma^2_{\pi_v}$) is not troublesome and does not make the proposed test less useful. Other considerations like correct size need to be considered when deciding upon a test. Figure 3.13 verifies that power of the proposed test to detect an extension alternative increased with the amount of the extension.

3.6 Discussion

This section discusses why, apart from the size and power considerations of Section 3.5, the proposed test based on t_p is a good test. In general these arguments revolve around the fact that design and population parameters are generally not known. Desirable and undesirable design characteristics when testing is a goal are discussed. What has been left undone by this report is also discussed.

It is clear from the simulations that sampling design can have a tremendous effect on the testing procedures. This observation is important because the temptation to ignore sampling design when making inferences exists and sometimes is large. The temptation to ignore sampling design is understandable because the effects on inference of unequal inclusion probabilities are often unknown and complicated. However, ignoring sampling design is a bad idea when two distribution functions are being

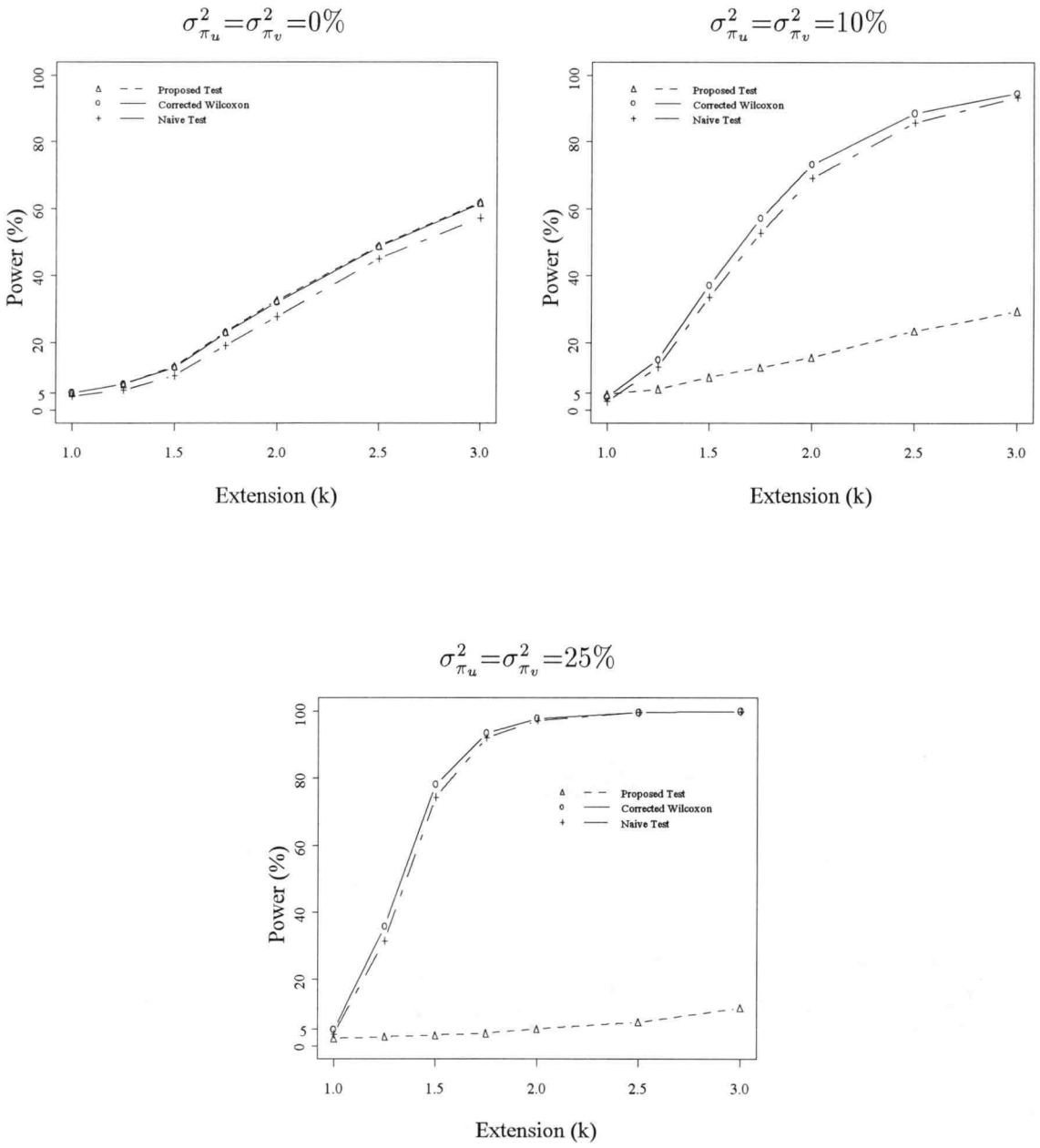

Figure 3.13: Power of the tests under *extension* alternatives. Sampling from Rentals population with g_u=g_v=10, n=m=27, and ρ_u=ρ_v=0.75.

compared. If the design is ignored, a person might believe that the size of a test is around 5% when in fact the actual Type I error rate is 100%. The sampling design *can* be ignored if simple random sampling or a design close to simple random sampling is performed from both populations. With simple random sampling, the only mistake in ignoring sampling design is loss of the correction to variance due to

the finite nature of the populations (i.e., the finite population correction factor). If inclusion probability variance is below 5% of its maximum and the same design is used in both populations ($\rho_u=\rho_v$), the simulations of Section 3.5 support the notion that design can be ignored.

The true rank correlation between responses and inclusion probabilities will not be known in most surveys prior to sampling when researchers are settling on a design. Sometimes this correlation can be estimated with auxiliary information or if pilot samples are available. The simulations in Section 3.5 suggest that this correlation can have a tremendous effect on the size of Wilcoxon's test. It seems reasonable to suggest that the proposed test based on t_p is the more useful test because it is less influenced by design correlation and inclusion probability variance and therefore can be applied with higher confidence when these quantities are unknown. Moreover, even when the corrected Wilcoxon procedure performs well (e.g., simple random sampling), the proposed procedure is virtually identical and therefore dominates the Wilcoxon at least on a subset of all possible designs and populations. True dominance of the proposed test cannot be established without exploring all possible designs and populations.

What design characteristics are good for hypothesis testing? First of all, using the same design in both populations assures that design correlations and inclusion probability variances are the same across populations (i.e., design is somewhere in the "trough" of Figure 3.5 through Figure 3.8) and the design can (effectively) be ignored provided inclusion probability variance is not too large. If variance in inclusion probabilities is low and the design is close to simple random, the simulations support the view that power is relatively high and decreases as the design moves away from simple random. That is, simple random sampling is probably the most powerful design. As always, increasing sample size increases power and it is desirable sample as many units as physically and financially possible. Increasing sample size also increases the accuracy of the distributional approximations. Undesirable design characteristics are basically the mirror-image of the desirable characteristics. Undesirable design char-

acteristics include inclusion probabilities which sort the two populations in different orders, inclusion probabilities with large variance, and small sample sizes.

Certain aspects of the general problem of hypothesis testing in finite populations have not been addressed in this paper, but will nonetheless be recorded here. Additional research needs to be focused on the case when populations are not the same size. The primary difficulty here is defining an appropriate null hypothesis. Eventually, researchers will want to compare more than two populations. The techniques given here need to be generalized to this situation. The ability to use different types of designs would be nice. Three important types of designs which need to be studied from a testing point of view are stratified designs, systematic designs, and non-fixed sample size designs.

3.7 Conclusions

This paper presented a rank-based testing procedure for comparing two finite population distribution functions defined on equal size populations. The test is based on a Horvitz-Thompson type correction of the usual Mann-Whitney U statistic and had approximately correct nominal size under a wide range of fixed-size randomized GRS designs. A corrected (for finiteness) form of the usual Wilcoxon rank sum statistic was also considered. The corrected Wilcoxon statistic, which ignores inclusion probabilities, had correct nominal size only under simple random sampling and exhibited extreme violations in size when design inclusion probabilities sorted the populations in opposite order or when the variance of inclusion probabilities got large. The proposed test was hypothesized to be a useful test because it displayed increasing power for certain alternatives and was relatively unaffected by changes in design correlation and inclusion probability variance. Simple random sampling was shown to be a good design for testing purposes because this design could effectively be ignored, thereby simplifying analysis, and was probably the most powerful design among those considered here. Designs with low inclusion probability variance are shown the be good designs for testing because of their similarities with simple random sampling.

Chapter 4

Summary

This chapter summarizes the entire thesis and discusses the need for further research in closely related topics.

This thesis explored two related topics in finite population statistics. Chapter 2 provided a relatively simple mechanism for estimating the sample size required to obtain, with specified probability, at least one member of some set of species. An "exact" solution to this problem was posed but shown to be computationally infeasible. Two approximate answers were then given and explored. Chapter 3 explores the complicated problem of testing for distributional differences in finite populations under general sampling designs. A goal throughout Chapter 3 was to expose the consequences of naively applying the regular two-sample Wilcoxon rank sum test when sampling designs are not simply random. An alternate test was proposed which overcame some of undesirable characteristics of the Wilcoxon test. The key feature of the alternate test was that it incorporated sampling weights by scaling individual Mann-Whitney U statistics the same way that the regular Horvitz-Thompson estimator scales observations.

The solutions proposed for both topics covered in this thesis could, in the correct setting, be applied to the same survey or population. This relationship between solutions is illustrated by the following hypothetical situation. Suppose scientists are interested in monitoring a certain segment of the environment for changes over time. The scientists might like to sample population under study in such a way that every species which has relative frequency over, say 20%, is represented in their sample. After measuring some characteristic (like gut content, weight, liver lesions, etc.) of individuals sampled at two different points in time, the scientists might like to compare distributions to assess whether or not characteristics have changed. The techniques of Chapter 2 can be applied to set the sample size while the techniques in Chapter 3 can be applied to test for distributional differences once the samples are drawn.

This thesis is closed with a listing of topics for additional research which might benefit the applied scientific community. Chapter 2 assumed multinomial sampling and, in a strict sense, requires that the probability of obtaining a species on any given trial does not change from trial to trial. This requirement is impossible to satisfy if the population being sampled is finite. It is likely that a hypergeometric sampling scheme will closely approximate reality when populations are finite and small. Additional research is needed into the probability mass function of the number of obtained species under hypergeometric sampling. Here, as in the multinomial case, exact computation is difficult and approximations are needed. A more sophisticated approximation (like Edgeworth or saddlepoint approximations) to the distribution used in Chapter 2 might also yield better performance in certain cases. In Chapter 3, the two populations under study were assumed to be of equal size. Additional research is needed to remove this assumption and propose a feasible test for distributional differences when populations are different sizes. Eventually, scientists will wish to compare more than two population CDF's simultaneously. With additional work the solutions in Chapter 3 might be extended to test for differences in more than two distribution functions.

BIBLIOGRAPHY

Brewer, K. R. W. and Hanif, M. (1983). *Sampling with unequal probabilities.* New York: Springer-Verlag.

Bunge, J. and Fitzpatrick, M. (1993). Estimating the number of species: A review. *Journal of the American Statistical Association, 88*(421), 364–373.

Cassel, C. M., Särndal, C. E., and Wretman, J. H. (1977). *Foundations of inference in survey sampling.* Wiley series in probability and mathematical statistics. New York: John Wiley and Sons.

Cochran, W. G. (1977). *Sampling Techniques* (3 ed.). Wiley series in probability and mathematical statistics. New York: John Wiley and Sons.

Horvitz, D. G. and Thompson, D. J. (1952). A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association, 47*, 663–685.

Johnson, N. L. and Kotz, S. (1977). *Urn Models and Their Application.* Wiley series in probability and mathematical statistics. New York: John Wiley and Sons.

Kaufmann, P. R., Herlihy, A. T., Elwood, J., Mitch, M., Overton, W. S., Sale, M. J., Messer, J. J., Cougan, K. A., Peck, D. V., Reckhow, K. H., Kinney, A. J., Christie, S. J., Brown, D. D., Hagley, C. A., and Jager, H. I. (1988). Chemical characteristics of streams in the mid-atlantic and southern united states. volumne i: Population descriptions and physico-chemical relationships. Technical Report EPA-600/3-88/021a, U.S. Environmental Protection Agency, Washington, DC.

Kish, L. (1965). *Survey Sampling.* New York: John Wiley and Sons.

Kolchin, V. F., Sevast'yanov, B. A., and Chistyakov, V. P. (1978). *Random Allocations.* Scripta Series in Mathematics. Washington, D.C.: V. H. Winston & Sons.

Kuk, A. Y. C. (1988). Estimation of distribution function and medians under sampling with unequal probabilities. *Biometrika, 75*, 97–103.

Lehmann, E. L. (1975). *Nonparametric statistical methods based on ranks.* San Francisco: Holden-Day.

Levene, H. (1960). Robust tests for equality of variances. In Olkin, I. (Ed.), *Contributions to Probability and Statistics*, pages 278–292. Stanford, Calif.: Stanford University Press.

Linthurst, R. A., Landers, D. H., Brakke, J. M., Overton, W. S., Meier, E. P., and Crow, R. E. (1986). Characteristics of lakes in the eastern united states. volume i: Population descriptions and physico-chemical relationships. Technical Report EPA-600/4-86/007a, U.S. Environmental Protection Agency, Washington, DC.

Madow, W. G. (1949). On the theory of systematic sampling ii. *Annals of Mathematical Statistics*, *20*, 333–354.

Mann, H. B. and Whitney, D. R. (1947). On a test of whether one of two random variables is stocastically larger than the other. *Annals of Mathematical Statistics*, *18*, 50–60.

Mood, A. M., Graybill, F. A., and Boes, D. E. (1974). *Introduction to the theory of statistics* (3 ed.). New York: McGraw-Hill.

Nath, H. B. (1973). Waiting time in the coupon-collector's problem. *Australian Journal of Statistics*, *15*(2), 132–135.

Nath, H. B. (1974). On the collector's sequential sample size. *Trabajos de estabdística*, *25*, 85–88.

Overton, W. S. (1985). A sampling plan for streams in the national stream survey. Technical Report 114, Oregon State University, Department of Statistics, Corvallis, OR.

Overton, W. S. (1986). Working draft analysis plan for the epa eastern lake survey. Technical Report 113, Oregon State University, Department of Statistics, Corvallis, OR.

Overton, W. S., Kanciruk, P., Hook, L. A., Eilers, J. M., Landers, D. H., Brakke, D. F., Blick, D. J., Linthurst, R. A., and DeHaan, M. D. (1986). Characteristics of lakes in the eastern united states. volume ii: Lakes sampled and descriptive statistics for physical and chemical variables. Technical Report EPA-600/4-86/007b, U.S. Environmental Protection Agency, Washington, DC.

Overton, W. S., White, D., and Stevens, D. L. (1990). Design report for emap: Environmental monitoring and assessment program. Technical Report EPA-600/3-90/053, U.S. Environmental Protection Agency, Washington, DC.

Rao, J. N. K. (1994). Estimating totals and distribution functions using auxiliary information at the estimation stage. *Journal of Official Statistics*, *10*, 153–165.

Rao, J. N. K., Kovar, J. G., and Mantel, H. J. (1990). On estimating distribution functions and quantiles from survey data using auxiliary information. *Biometrika*, *77*, 365–375.

Rao, J. N. K. and Scott, A. J. (1984). On chi-squared tests for multiway contingency tables with cell proportions estimated from survey data. *Annals of Statistics*, *12*(1), 46–60.

Rao, J. N. K. and Scott, A. J. (1987). On simple adjustments to chi-square tests with sample survey data. *Annals of Statistics*, *15*(1), 385–397.

Riordan, J. (1968). *Combinatorial Identities*. Wiley series in probability and mathematical statistics. New York: John Wiley and Sons.

Sale, M. J., Kaufmann, P. R., Jager, H. I., Coe, J. M., Cougan, K. A., Kinney, A. J., Mitch, M. E., and Overton, W. S. (1988). Chemical characteristics of streams in the mid-atlantic and southern united states. volumne ii: streams sampled, descriptive statistics, and compendium of physical and chemical data. Technical Report EPA-600/3-88/021b, U.S. Environmental Protection Agency, Washington, DC.

Särndal, C. E., Swensson, B., and Wretman, J. (1992). *Model assisted survey sampling*. New York: Springer-Verlag.

Sedransk, N. and Sedransk, J. (1979). Distinguishing among distributions using data from complex sample designs. *Journal of the American Statistical Association*, *74*, 754–776.

Sokal, R. R. and Rohlf, F. J. (1981). *Biometry* (2 ed.). San Francisco: W. H. Freeman and Company.

Stehman, S. V. and Overton, W. S. (1994a). Comparison of variance estimators of the horvitz-thompson estimator for randomized variable probability systematic sampleing. *Journal of the American Statistical Association*, *89*(425), 30–43.

Stehman, S. V. and Overton, W. S. (1994b). Environmental sampling and monitoring. In Patil, G. P. and Rao, C. R. (Eds.), *Handbook of Statistics*, volume 12. Elsevier Science B. V.

Sunter, A. (1986). Solutions to the problem of unequal probability sampling without replacement. *International Statistical Review*, *54*, 33–50.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, *1*, 80–83.

Zar, J. H. (1974). *Biostatistical Analysis*. Biological sciences series. Englewood Cliffs, N.J.: Prentice-Hall.

APPENDICES

Appendix A

The Occupancy Probability Function

Let $\mathbf{Y}$ be the (random) number of species, out of k, observed at completion of n draws. Assume that the draws are independent and that the probability of obtaining a member of species i ($i = 1, \ldots, k$) does not change in the course of drawing. We seek functions which yield $Pr(\mathbf{Y} = y)$ (called the *pmf* or "probability mass function") and $Pr(\mathbf{Y} \leq y)$ (called the *cmf* or "cumulative mass function").

Both Johnson and Kotz (1977) and Nath (1974) give equivalent expressions for the *pmf* presented below. However, we believe the existence of the *pmf* is not well known and therefore warrants presentation in yet another form.

Let the vector $(n_{o_1}, n_{o_2}, \ldots, n_{o_k})$ be a realization from a k-dimensional multinomial distribution having n trials. ($0 \leq n_{o_i} \leq n$ for all $i = 1, \ldots, k$). Let $I(e)$ be the indicator function for the event e. $\mathbf{Y}$ can now be defined as $\mathbf{Y} = \sum_{i=1}^{k} I(n_{o_i} \geq 1)$, the number of n_{o_i} which are greater than or equal to one. Letting $\pi_{(i)}$ be the probability of obtaining a member of species i on any draw, the *pmf* of $\mathbf{Y}$ is,

$$Pr(\mathbf{Y} = y) = \sum_{(n_{o_1}, n_{o_2}, \ldots, n_{o_k}) \in \psi_y} \binom{n}{n_{o_1} n_{o_2} \ldots n_{o_k}} \left\{ {\pi_{(1)}}^{n_{o_1}} {\pi_{(2)}}^{n_{o_2}} \ldots {\pi_{(k)}}^{n_{o_k}} \right\} \tag{A.1}$$

where ψ_y represents the set of all k-tuples of non-negative integers $(n_{o_1}, n_{o_2}, \ldots, n_{o_k})$ such that $\sum_i n_{o_i} = n$ and $\sum_i I(n_{o_i} \geq 1) = y$.

We now simplify Equation A.1. Let the set ζ_j contain all subsets of size j from the integers $\{ 1, 2, \ldots, k \}$. There are $\binom{k}{j}$ unordered sets in ζ_j. We now define

$$Q_j = \sum_{s \in \zeta_j} {\omega_s}^n. \tag{A.2}$$

where $\omega_s = \sum_{i \in s} \pi_{(i)}$. Note that $Q_k \equiv 1$, and that $Q_{k-j} = \sum_{s \in \zeta_j} (1 - \omega_s)^n$, the latter of which tell us that subsets of $\{ 1, \ldots, k \}$ (i.e., the ζ_j) only need to be enumerated up to size $[k/2] + 1$, the smallest integer greater than $k/2$.

The *pmf* of $\mathbf{Y}$ is

$$Pr(\mathbf{Y} = y) = \sum_{j=1}^{y} (-1)^{y-j} \begin{pmatrix} k-j \\ y-j \end{pmatrix} Q_j \tag{A.3}$$

This expression is equivalent to equation 3.5, page 109 of Johnson and Kotz (1977).

In most instances, it is convenient to also have an expression for the cumulative mass function. Using problem 9, page 33 of Riordan (1968) to re-express the coefficients of Q_j, we obtain

$$Pr(\mathbf{Y} \leq y) = \sum_{j=1}^{y} Pr(\mathbf{Y} = j) = \sum_{j=1}^{y} (-1)^{y-j} \begin{pmatrix} k-j-1 \\ y-j \end{pmatrix} Q_j \tag{A.4}$$

Equation A.4 is equivalent to the equation at the top of page 157 of Johnson and Kotz (1977).

A final note. Duplicate values of $\pi_{(i)}$ can sometimes allow significantly faster computation. When several of the $\pi_{(i)}$ are equal, some of the ω_s are equal and we need only enumerate those subsets which produce different ω_s. Suppose there are only k_d distinct values which the $\pi_{(i)}$ $(i = 1, \ldots, k)$ take on. Let $\pi_{d_{(h)}}$ represent the h-th distinct value taken on by the $\pi_{(i)}$ $(h = 1, \ldots, k_d)$ and assume that there are r_h of the $\pi_{(i)}$ which equal $\pi_{d_{(h)}}$. Given a set of non-negative integers i_1, i_2, ..., i_{k_d} summing to j, there are $\prod_{h=1}^{k_d} \binom{r_h}{i_h}$ identical ω_s's in Equation A.2, each of which is equal to $(i_1\pi_{d_{(1)}} + i_2\pi_{d_{(2)}} + \ldots + i_{k_d}\pi_{d_{(k_d)}})$. By summing over all size k_d sets of non-negative integers summing to j (i.e., all "k_d part partition of j"), we can re-express Q_j as,

$$Q_j = \sum_{i_1+\ldots+i_{k_d}=j} \binom{r_1}{i_1} \cdots \binom{r_{k_d}}{i_{k_d}} (i_1\pi_{d_{(1)}} + \ldots + i_{k_d}\pi_{d_{(k_d)}})^n \tag{A.5}$$

where, as usual, $\binom{r_h}{i_h} = 0$ when $i_h > r_h$. Note that the number of terms in Equation A.5 depends on the number of distinct $\pi_{(i)}$ (i.e., k_d) and multiplicities r_i rather than directly on k.

Intermediate computations illustrating calculation of a single point in the *cmf* when k=4, n=5, and $(\pi_{(1)}, \pi_{(2)}, \pi_{(3)}, \pi_{(4)})$ =(0.2, 0.2, 0.3, 0.3) appear in Table A.1. Here k_d=2, r_1=2, r_2=2, $\pi_{d_{(1)}}$=0.2, and $\pi_{d_{(2)}}$=0.3.

Relevant Subsets:	Via Equation A.2:
$\zeta_1 = \{ \{ 1 \}, \{ 2 \}, \{ 3 \}, \{ 4 \} \}$	$Q_1 = 0.2^5 + 0.2^5 + 0.3^5 + 0.3^5 = 0.03302$
$\zeta_2 = \{ \{ 1,2 \}, \{ 1,3 \}, \{ 1,4 \}, \{ 2,3 \},$	$Q_2 = (0.2 + 0.2)^5 + (0.2 + 0.3)^5 +$
$\{ 2,4 \}, \{ 3,4 \} \}$	$(0.2 + 0.3)^5 + (0.2 + 0.3)^5 +$
	$(0.2 + 0.3)^5 + (0.3 + 0.3)^5 = 0.52412$
Via Equation A.5:	
$Q_1 = \binom{2}{1}\binom{2}{0}(0.2)^5 + \binom{2}{0}\binom{2}{1}(0.3)^5$	Using Equation A.4:
$= 0.03302$	$\Pr(\mathbf{Y} \leq 2) = (-1)\binom{2}{1}Q_1 + \binom{1}{0}Q_2$
$Q_2 = \binom{2}{2}\binom{2}{0}(2(0.2))^5 +$	$= 0.4581$
$\binom{2}{1}\binom{2}{1}(0.2 + 0.3)^5 +$	
$\binom{2}{0}\binom{2}{2}(2(0.3))^5 = 0.52412$	

Table A.1: Calculating $Pr(\mathbf{Y} \leq 2)$ when k=4, n=5, and the probabilities of each species are (0.2, 0.2, 0.3, 0.3). Here k_d=2, r_1=2, r_2=2, $\pi_{d_{(1)}}$=0.2, and $\pi_{d_{(2)}}$=0.3.

Appendix B

Generalized Random Samples

Simulations conducted in Chapter 3 drew fixed-size *grouped* randomized general random samples (grouped GRS). Designs which can be generated as grouped GRS include simple random sampling (SRS) (Cochran, 1977), randomized variable probability sampling (randomized VPS) with probability proportional to x (πpx) (Stehman and Overton, 1994a), and various "plotless" designs which do not require a sampling frame nor *a priori* knowledge of inclusion probabilities. Designs which cannot be generated as randomized GRS samples include systematic designs and stratified designs, both of which restrict randomization of the GRS. Second order inclusion probabilities for randomized GRS samples are not generally known because of computational difficulties. A good approximation of second order inclusion probabilities in a randomized GRS is given in Overton (1985) and appears in Chapter 3 on page 41.

Grouped GRS define a certain number of distinct first order inclusion probabilities for the population. Thus, samples are drawn such that units within a "group" are sampled with the equal inclusion probability. Units in different groups are sampled with different inclusion probabilities. For example, there may be 5 units sampled with π_1 probability, 10 units sampled with π_2, 20 with π_3, etc. If each unit is sampled with a unique probability (i.e., groups are of size 1), the inclusion probabilities are typically proportional to a auxiliary variable (x), but this is not necessary.

To illustrate the physical drawing of a grouped GRS sample, consider the following example. Suppose a sample of two units from a population of five units, labeled u_1 through u_5, is desired. Suppose the inclusion probability vector for the units is $\boldsymbol{\pi_u}$=[0.3, 0.3, 0.3, 0.55, 0.55]$'$, where the first element in $\boldsymbol{\pi_u}$ is π_{u_1}, the second element is π_{u_2}, and so on. Note that in fixed size designs it is always true that $0 < \pi_i \leq 1$ and $\sum_i \pi_i = n$. Having set inclusion probabilities according to the design, the order of $\boldsymbol{\pi_u}$ is randomized. Suppose this randomization results in the order vector $o = [5, 3, 1, 4, 2]'$ corresponding to $\boldsymbol{\pi_u}[o] = [\pi_{u_5}, \pi_{u_3}, \pi_{u_1}, \pi_{u_4}, \pi_{u_2}]' = [0.55, 0.3, 0.3, 0.55, 0.3]'$. Let o_i be the i-th index in the random order vector o. "Break points" are now constructed as

$b_i = \sum_{k=1}^{i} \pi_{u_{o_k}}$ for i=0,1,...,5. In this example the break points are $b_0 = 0$, $b_1 = 0.55$, $b_2 = 0.85$, $b_3 = 1.15$, $b_4 = 1.7$, $b_5 = 2$. The intervals $[b_{i-1}, b_i]$ for $i = 1, \ldots, n$ are then associated with the units $u_{o_{i+1}}$. In this example, $[b_0,b_1]$ is associated with u_5, $[b_1,b_2]$ is associated with u_3, and so on. A random number from a Uniform[0,1] distribution is drawn, call this number p, and the units associated with the intervals containing $p + k$ for k =0, 1, ..., (n-1) are taken in the sample. Suppose the random draw from a Uniform[0,1] results in $p = 0.635$. Units u_3 and u_4 are taken in the sample because 0.635 falls in the interval $[b_1,b_2]$ associated with u_3 and 1.635 falls in the interval $[b_3,b_4]$ associated with u_4.

Appendix C

Listing of Computer Code

This appendix lists the source code used to perform the simulations presented in Section 3.4. The simulations were implemented using SAS© (SAS Institute, Cary NC) and relied heavily on the SAS macro facility and Proc IML, the matrix procedure in SAS. It is the intent of this appendix to provide a permanent record of the source code of the simulation. The author will give electronic copies of the simulation files away freely. A reader who is interested in obtaining an copy should first attempt to contact the author.

The structure of the simulation program is modular with routines stored in separate files and "compiled" before the actual simulations are run. Below, each file is delineated with a comment line looking something like `/* ######## */`. Each of these files should be executed prior to the simulation so that the routines are available at simulation run time as object code. SAS version 6 and above have the ability to store IML object code in a SAS catalog. It is unknown whether SAS versions prior to 6 support this capability. Un-edited comments appear throughout the listing to (hopefully) aid interpretation.

The contents of file LIBREF.SAS:

```
%let droot=c:\trent\ranksamp\sassimu;
libname procs "&droot\procs";
libname results "&droot\results";
libname simu "&droot";
```

The simulation files:

```
/*      SIMULATION PARAMETER FILE.

        This file is the one actually submitted to SAS for execution.
*/
options nosource2 nomprint nosymbolgen;
proc printto log="simulog.log";
/* Root directory for storage */
%let simuroot=c:\;
/* Libraries. All modules are stored in catalogs in library PROCS.
```

```
      All results are stored in files in library RESULTS. */
%include "&simuroot\librefs.sas";
/* Macro to open the correct storage catalogs where all modules
   can be found */
%let opencats=%quote(reset storage=procs.gprocs;
   load module=(DIAGRV OVERPIJ SAMPLE SCALEPI SCALEX2
   rankcor setcor  SORTMAT VAR VEC VECVAR);
   reset storage=procs.simprocs;
   load module=(DISPITER EZVAR GRS grouppi2 lakes
   normpop kish PROP1E SETRAND STORITER
   STORSIMU SUMSIMU TESTP1); );
/* ------------------------------------------------ */
%macro setlpop1;
    call lakes(pop1, area, bign);
%mend;
/* ------------------------------------------------ */
%macro setlpop2;
    call lakes(pop2, area, bigm);
%mend;
/* ------------------------------------------------ */
%macro setkpop1;
    call kish(pop1, dwells, bign);
%mend;
/* ------------------------------------------------ */
%macro setkpop2;
    call kish(pop2, dwells, bigm);
%mend;
/* ------------------------------------------------ */
%macro setnpop1;
        call normpop(pop1, bign);
%mend;
/* ------------------------------------------------ */
%macro setnpop2;
        call normpop(pop2, bigm);
%mend;
/* ------------------------------------------------ */
%macro initmac;
file log;
        pivec1 = grouppi2( &n, bign, &pctvar1, &ngroup);
        call setcor( p, truer1, pivec1, pop1, &cor1, 0.01 );
        pivec1 = scalepi( &n, pivec1 );
        pivec2 = grouppi2( &m, bigm, &pctvar2, &ngroup);
        call setcor( p, truer2, pivec2, pop2, &cor2, 0.01 );
        pivec2 = scalepi( &m, pivec2 );
        call ezvar(maxapv, avghtv, avgapv, pivec1, pivec2 );
```

```
        avgapv = avgapv / (bign*bigm)**2;
        avghtv = avghtv / (bign*bigm)**2;
%mend;
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/*       MACRO TO SET PARAMETERS AND RUN SIMULATION              */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
%MACRO GOSIMU( SSIZE1, SSIZE2, SHFT, EXT, PCTVAR1, PCTVAR2, COR1,
COR2, NGROUP, POPLN1, POPLN2, RESULTNM, RSEED );
/*
        Purpose: To set macro variables (parameters) and call MAIN
                 controling simulation procedure.
        Inputs: SSIZE1 = size of first sample
                SSIZE2 = size of second sample
                SHFT = a set of shifts for alternative, 0=Null
                EXT =  a set of extentions for alternative, 1 = null
                PCTVAR1, PCTVAR2 = % of max variance of pi
                COR1, COR2 = desired rank corr btwn incl probs and
     responses
                NGROUP = Number of groups in pi vectors
                POPLN1, POPLN2 = Populations to sample from
                RESULTMN = name of results catalog
                RSEED = the random number seed (stored with results)
*/
%let n=&SSIZE1;
%let m=&SSIZE2;
%let shift=&SHFT;
/* Define some statements which get executed once at the begining of
   the simulation.  THIS MACRO VARIABLE IS NOT SAVED IN CATALOG, and
   therefore can be as long as needed. (ie. more than 200 characters )
*/
%let initstat=%quote( %initmac );
/* Define routines to take the samples */
%let design1=grs(s1, pi1, pop1, &n, pivec1);/* output is s1 and pi1 */
%let design2=grs(s2, pi2, pop2, &m, pivec2);/* output is s2 and pi2 */
/* Define routine to compute statistics every iteration */
/* Output from PROP1:
        t1      = sum of z using (bign*bigm) as denom
        t2      = sum of z using sum(1/piu*piv) as denom
        rs      = usual rank sum = n*m + n(n+1)/2 - nm/(NM)^2*t1
   t1, t2, and rs are vectors with the statistics corresponding to
   the shifts AND EXTENTIONS present in the vector shift AND EXTEND.
   Structure of result is
| Columns for t1        | Columns for t2        | Columns for rs        |
| shift t1 | extend t1 | shift t2 | extend t2 | shift rs | extend rs |
*/
```

```
%let estimate=prop1e(t1, t2, rs, s1, pi1, s2, pi2, bign, bigm,
    &shift, &ext);
/* Define routine to summarize every iteration (ITERSUM) and entire
   simulation (SIMUSUM) */
%let alpha = 0.05;
%let itersum=testp1(t1 );
%let iresmat=iterres;
%let sresmat=simures;
%let srownm=sresrow;
/* Output is &sresmat and &srownm, summary of iteration results
   are put in &sresmat */
%let simusum=sumsimu(&sresmat, &srownm, &iresmat);
/* Name of the iteration variable.  This variable is incremented
   once each iteration. */
%let itervar=iter;
/* Diplay routine */
/* Must use single ' in call for storage to work out */
%let dispit=dispiter( &itervar || t1[1,1:2] || t2[1,1:2]||rs[1,1:2],
                      {'Iter' 'T10' 'T1a' 'T20' 'T2a' 'RS0' 'RSa' } );
/* Routine to set the random number seed */
%let seed=&RSEED;
%let setseed=setrand( &seed );
/* Function to set the number of iterations */
%let numiter=2000;
/* ------------ Storage considerations---------- */
/* Root name for all results pertaining to this simulation */
%let nameroot=&RESULTNM;
/* The number of iterations between writing results to disk */
%let numpsave=10;
/* Storing iteration results:
  Variables named in &isavlist are appended to the bottom of a matrix
  named &iresmat. Labels for the columns of &iresmat are given in
  &isavnm and are stored in the catalog as a string matrix named
  &icolnm. */
%let icolnm=irescol; /* name of name matrix */
%let isavlist=(t1 || t2 || rs);
%let isavnm={'Shift t1' 'Extend t1' 'Shift t2' 'Extend t2'
'Shift rs' 'Extend rs'};
%let istor=%quote( %storit( &numpsave) );
/* Call to store final iteration results */
%let fistor=%quote( %storit( 1 ) );
/* Storing simulation results:
 Variables named in &ssavlist are stored in a matrix called &sresmat.
 Labels for the columns of &sresmat are listed in &ssavnm and stored
 in matrix &scolnm.  Labels for the rows of &sresmat are stored in
```

```
 the character matrix &srownm and are assigned in the procedure
 SUMSIMU. */
%let ssavnm=&isavnm;
%let ssavlist=( &sresmat  );
%let scolnm=srescol;
%let sstor=%quote( %storsi );
/* Call to store any more simulation quantities (like) parameters */
/* parstor stores all macro variable in %let statements into catalog.
The list of variables to store is in parstor*/
%let fsstor=%quote( %parstor );
/* ---------------- include the controling procedure ------------ */
%include "&simuroot\main.sas";
%mend GOSIMU;
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
/*    EXAMPLE CALLS TO GOSIMU WHICH INVOKE A SIMULATION           */
/* ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++ */
* This one has n=m=27, no shift or extension alternative to simulate,
  variance in pi vector is 25% from both pops, rank correlation is
  1 in pop1, -0.5 in pop2, 5 groups in design, and sampling is
  from Rental (kish) population.  Results are stored in the catalog
  EXAMP1. Seed is 1234 ;
%gosimu( 27, 27, {0},{1}, 0.25, 0.25, 1.0,-0.5, 5,
   %quote(%setkpop1), %quote(%setkpop2), examp1, 1234);

*  This one has n=m=27, no shift, shift of 50, and extension
   alternative to simulate, variance in pi vector is 0 (SRS) from
   Population 1, 30% from Population 2, rank correlation is .5 in
   pop1, 0 in pop2, 5 goups in design, and sampling is from Rental
   (kish) population.   Results are stored in the catalog EXAMP2.
   Seed is 5678 ;
%gosimu( 27, 27, {0 50},{1}, 0.0, 0.3, 0.5, 0.0, 5,
        %quote(%setnpop1), %quote(%setnpop2), examp2, 5678);
/* ############################################################## */
/* ------------------------------------------------------------
        MAIN.SAS
        Main include file for my simulation. This file is included
        by the parameter file an is the controling file for the
        simulation.
   ------------------------------------------------------------ */
/* macro var simuroot defined in parameter file */
%include "&simuroot\librefs.sas";
proc iml symsize=150;
start main;
/*
   MAIN Module.
```

```
   Purpose: Control execution of the simulation.
            It goes roughly as follows.
            Two populations read, two samples are drawn,
            a statistic is computed,
            statistic(s) is stored, and the process is iterated.
   All maticies defined in module main are global because main
   has no parameters.
*/
        /* Set the populations. These macro variables contain
           calls to the correct procedures.
           Macro variables are defined in the parameter file. */
        &popln1;
        &popln2;
        /* Initialize other variables */
        &initstat;
        /* Set the random number sequence */
        call &setseed;
        /* This is the iteration loop */
        do &itervar = 1 to &numiter;
                /* Sample according to the designs */
                call &design1;
                call &design2;
                /* Compute the statistic(s) */
                call &estimate;
                /* Compute any summaries for each iteration */
                call &itersum;
                /* Display some results */
                call &dispit;
                /* Store iteration results */
                &istor;
        end;
        /* Store final iteration results */
        &fistor;
        /* Compute any simulation summaries */
        call &simusum;
        /* Store overall simulation results */
        &sstor;
        /* Store simulation parameters */
        &fsstor;
finish;
/*  Open storage catalogs and load modules */
&opencats;
call main;  /* go */
quit;
/* ############################################################## */
```

```
/*
        KISH.SAS
        Read the data set from Kish (1965) about Rentals
        on blocks in a town in Mass.
*/
%include "c:\librefs.sas";
proc iml;
start kish(rentals, dwell, bign);
/*
        Purpose: to read the Kish data.
        Input: none
        Output:
          rentals = "y" in Kish = number of rentals on block
          dwell = "x" in Kish = number of houses/dwellings on block
          bign = number of blocks in data set
*/
* Order of columns goes Y=Rentals X=Dwellings, Y, X, etc
 Y,      X,      Yj      Xi        Y,      X;
xy = {
 131     149     30      43        2       5
 6       10      41      53        5       11
 23      30      37      47        6       14
 79      90      12      18        13      29
 47      56      25      41        28      42
 34      42      23      33        27      36
 97      113     26      39        22      30
 30      45      50      67        4       11
 11      16      31      41        10      19
 45      67      24      35        25      42
 19      23      47      53        67      110
 17      18      27      28        44      57
 25      33      80      90        43      81
 84      89      52      68        15      23
 91      114     90      99        17      25
 48      66      78      89        29      59
 48      61      46      48        18      27
 20      25      35      48        14      22
 34      46      59      62        24      29
 42      58      27      33        35      44
 35      44      33      43        48      53
 55      66      27      37        20      27
 42      61      9       14        24      28
 36      45      9       15        55      62
 13      20      12      21        43      56
 7       16      49      68        13      22
```

8	15	60	81	19	22
18	26	35	59	48	57
20	22	11	23	44	57
18	22	21	32	36	46
0	2	22	36	3	8
23	29	10	16	2	4
0	3	9	15	13	18
19	29	7	16	34	42
11	21	3	8	28	32
11	15	5	25	23	28
42	54	2	11	8	14
28	42	8	9	69	76
8	13	14	19	0	19
0	2	5	5	5	9
34	48	1	3	6	37
13	24	22	37	4	11
16	27	25	30	9	24
21	32	2	3	54	102
12	14	0	4	50	82
10	18	7	13	9	24
50	61	15	24	6	18
58	65	10	19	5	18
17	25	5	17	1	3
41	68	8	13	0	6
3	8	8	18	0	1
4	12	0	1	2	7
18	27	4	10	2	8
1	3	1	4	3	12
1	3	3	9	0	4
3	6	0	5	6	8
6	14	14	20	3	9
5	15	3	5	3	7
5	14	5	13	5	12
4	9	0	1	3	10
0	1	11	23	0	1
0	4	19	39	0	1
7	12	5	9	0	1
7	22	0	2	2	4
3	11	3	5	0	1
12	27	12	26	0	1
11	20	4	10	0	2
27	38	14	35	0	1
14	31	0	4	0	1
2	4	20	38	0	1
3	23	8	10	0	1

```
 6        12       22       36        0        2
 5        15       2        7         0        2
 31       39       2        4         0        4
 7        9        0        2         0        2
 3        11       1        3         0        2
 1        10       17       29        0        4
 3        8        24       44        0        1
 1        3        3        5         0        1
 6        10       7        17        3        5
 4        8        0        2         4        9
 5        12       18       42        0        1
 1        3        2        4         0        2
 0        1        0        1         0        2
 0        3        0        1         2        6
 8        22       0        3         1        4
 8        18       0        1         3        7
 16       25       5        14        11       14
 9        25       0        1         3        9
 6        20       0        1         0        2 };
xy = shape(xy, 270, 2);
rentals = xy[,1];
dwell = xy[,2];
bign = nrow(rentals);
finish;
reset storage=procs.simprocs;
store module=kish;
quit;
/* ################################################################ */
/*  ---------------------------------------------------------
        LAKES.SAS
        IML routine to get the LAKES data set.
    ---------------------------------------------------------- */
%include "c:\librefs.sas";
proc iml;
start lakes(anc, area, bign);
/*
   Purpose: to get and return the New England Lakes data set.
      The dataset consists of ANC (Acid Nutilizing Capacity) and
      AREA (hectares) measured on 110 lakes in New England.
   Inputs: none
   Outputs: anc = 110X1 vector of anc values
            area = 110X1 vector of area values
            bign = scalar equal to 110, the population size
*/
ancarea={
```

-31.300	14.000
-8.1000	22.000
-6.5000	35.000
-3.8000	112.00
-0.10000	33.000
0.50000	13.000
1.0000	21.000
3.4000	16.000
4.0000	210.00
4.8000	4.0000
5.0000	8.0000
5.5000	11.000
6.0000	18.000
7.0000	11.000
7.0000	21.000
7.9000	29.000
8.0000	11.000
8.0000	5.0000
9.0000	15.000
9.3000	25.000
10.000	26.000
10.200	100.00
10.300	97.000
10.800	37.000
12.000	4.0000
12.000	39.000
12.100	5.0000
12.300	9.0000
12.500	37.000
13.000	9.0000
15.200	49.000
15.300	9.0000
17.000	5.0000
19.700	5.0000
20.000	8.0000
21.000	30.000
22.500	83.000
23.000	12.000
24.000	495.00
24.800	4.0000
24.900	46.000
26.000	18.000
30.000	6.0000
32.000	17.000
32.800	4.0000

35.000	6.0000
36.000	34.000
39.800	7.0000
40.500	7.0000
41.000	7.0000
52.000	6.0000
54.000	17.000
56.000	8.0000
63.000	12.000
67.000	14.000
68.000	13.000
68.900	15.000
69.300	6.0000
71.000	3.0000
76.900	9.0000
78.500	12.000
80.000	44.000
83.000	53.000
83.000	6.0000
86.000	36.000
101.00	17.000
102.00	18.000
117.50	14.000
138.00	57.000
140.50	9.0000
142.00	14.000
152.00	21.000
168.00	57.000
192.20	6.0000
213.00	84.000
254.00	5.0000
262.00	274.00
262.00	55.000
292.00	5.0000
296.00	84.000
302.00	40.000
320.10	9.0000
320.10	9.0000
333.00	137.00
357.00	89.000
372.00	123.00
373.00	162.00
377.00	81.000
396.00	4.0000
421.40	8.0000

```
  460.00   233.00
  477.00   65.000
  584.40   144.00
  593.00   13.000
  608.00   395.00
  626.50   1213.0
  652.20   430.00
  682.00   253.00
  732.50   10.000
  756.40   284.00
  783.50   106.00
  795.00   81.000
  804.00   10.000
  806.00   47.000
  854.00   4.0000
  971.00   19.000
  971.00   19.000
  1096.4   73.000
  1807.5   22.000
  2123.7   13.000};
ancarea=shape(ancarea,110,2);
anc = ancarea[,1];
area = ancarea[,2];
bign = nrow(anc);
finish;
reset storage=procs.simprocs;
store module=lakes;
quit;
/* ################################################################ */
/*  ---------------------------------------------------------
        NORMPOP.SAS
        IML routine to get the NORMAL data set.
    ---------------------------------------------------------- */
%include "c:\librefs.sas";
proc iml;
start normpop(x,  bign);
/*
        Purpose: To return a set of randomly generated
           deviates from a normal distribution.
        Inputs: none
        Outputs: x  = (bign)X1 vector of normal deviates
                 bign = scalar equal to 500, the population size
*/
x={
 117.0, 72.3, 118.7,  70.0, 117.5,  84.4,  88.8,  98.4, 110.1, 138.4,
```

107.4, 91.5, 92.3, 92.3, 69.7, 110.1, 155.2, 89.3, 89.9, 105.5,
98.9,129.6, 74.1, 133.2, 103.0, 108.5, 121.9, 134.2, 71.7, 98.9,
79.3,110.4, 43.6, 101.6, 93.7, 104.3, 122.8, 126.7, 136.9, 74.6,
62.0, 98.7, 144.3, 83.0, 100.6, 97.7, 111.8, 94.9, 87.2, 67.4,
116.6, 90.5, 55.7, 75.1, 125.5, 107.7, 92.6, 130.6, 106.3, 104.4,
85.2,128.7, 116.9, 129.1, 80.1, 81.4, 107.8, 111.8, 109.2, 127.9,
58.4, 79.9, 116.7, 76.5, 92.5, 71.0, 106.0, 83.9, 50.4, 124.9,
77.7, 27.6, 70.0, 100.8, 93.3, 105.3, 75.8, 83.2, 102.7, 81.7,
80.2, 92.9, 112.5, 109.2, 113.7, 121.2, 51.2, 65.6, 100.2, 122.5,
90.6,104.4, 143.1, 121.7, 98.0, 84.6, 42.1, 73.2, 161.6, 99.8,
120.4, 56.3, 127.0, 44.3, 156.4, 29.4, 69.9, 110.6, 94.6, 112.9,
117.4, 87.7, 91.4, 121.2, 93.8, 69.6, 113.4, 77.2, 108.6, 111.6,
87.9,119.6, 103.8, 133.0, 22.5, 96.8, 103.3, 128.1, 75.8, 72.0,
88.6, 81.7, 91.2, 62.8, 82.1, 98.7, 95.5, 85.3, 135.2, 111.1,
65.8, 44.5, 106.8, 112.9, 101.7, 108.9, 55.2, 53.2, 43.0, 106.5,
100.6,110.1, 92.0, 123.1, 95.8, 92.7, 136.2, 103.8, 82.9, 107.5,
115.4, 90.6, 80.1, 105.6, 56.1, 63.2, 108.7, 114.2, 89.0, 103.4,
104.2, 62.1, 118.0, 80.9, 80.0, 89.7, 114.2, 85.4, 80.8, 92.9,
123.0,133.3, 114.6, 96.3, 107.6, 144.3, 69.2, 108.8, 63.3, 78.4,
104.1,117.1, 91.1, 67.6, 121.7, 91.8, 86.3, 84.1, 53.3, 109.9,
122.1, 82.5, 104.1, 113.3, 110.2, 112.7, 104.4, 94.5, 87.3, 70.5,
119.6,112.3, 86.3, 95.8, 111.5, 59.7, 109.5, 94.4, 105.7, 143.8,
123.5, 57.2, 105.3, 75.5, 94.9, 55.4, 157.8, 104.4, 113.8, 111.0,
85.8, 83.1, 48.9, 69.7, 101.3, 87.7, 85.3, 113.4, 38.4, 147.7,
111.5, 97.4, 136.2, 117.4, 89.2, 89.2, 129.2, 134.8, 104.2, 87.7,
86.4, 95.3, 107.9, 115.5, 107.0, 132.9, 71.0, 92.3, 68.8, 102.1,
79.1,116.7, 81.3, 101.3, 140.9, 114.2, 90.3, 85.1, 99.7, 133.7,
100.0,115.0, 117.5, 103.8, 97.8, 106.5, 86.8, 88.3, 44.4, 115.8,
90.8,103.2, 80.0, 90.9, 89.7, 71.1, 86.8, 97.9, 68.7, 47.5,
49.7, 55.3, 130.9, 123.7, 95.6, 116.4, 68.1, 98.7, 112.7, 89.9,
53.7, 31.7, 40.2, 166.4, 83.3, 69.0, 100.9, 90.8, 95.2, 58.0,
90.6,120.9, 63.4, 99.6, 111.6, 92.7, 90.4, 111.0, 136.0, 110.2,
103.8,124.1, 101.5, 99.8, 76.4, 45.2, 116.0, 69.0, 135.2, 125.5,
112.0,105.5, 117.9, 104.1, 116.3, 71.6, 62.8, 72.5, 91.4, 111.0,
100.9,104.6, 149.3, 109.5, 123.0, 99.2, 106.8, 103.1, 108.4, 106.4,
124.3, 91.1, 126.3, 109.3, 62.7, 121.5, 127.0, 103.6, 91.6, 74.6,
96.6, 98.5, 84.1, 115.8, 127.5, 115.9, 81.5, 66.7, 72.9, 88.7,
110.9,100.3, 88.9, 103.3, 104.2, 161.0, 119.1, 144.8, 107.4, 145.7,
114.7,113.5, 99.5, 80.2, 90.0, 59.8, 95.0, 73.5, 91.1, 81.8,
111.8,108.3, 95.9, 66.0, 92.9, 125.9, 87.8, 94.5, 91.4, 90.3,
96.3,136.1, 83.6, 99.1, 124.2, 130.4, 84.0, 49.6, 91.3, 75.1,
105.4,132.2, 81.9, 132.1, 109.4, 79.0, 68.2, 62.9, 97.1, 64.0,
74.3,112.7, 152.2, 142.5, 73.8, 103.9, 136.8, 127.4, 137.6, 94.3,
98.0, 82.5, 78.0, 152.1, 107.4, 129.9, 110.9, 99.9, 97.6, 68.3,
114.2, 98.5, 156.1, 83.2, 82.9, 71.0, 122.4, 111.6, 98.0, 108.7,

```
 136.8,179.7, 123.3,  76.6,  64.0,  89.6,  80.8,  86.8, 112.3, 141.0,
  57.7,116.4,  99.5, 113.2,  73.7,  80.2, 131.9, 138.4, 161.0, 102.4,
 139.3, 61.1,  66.7,  90.8, 103.3,  96.1,  98.7,  86.0, 139.0, 121.3,
 104.7,132.4, 143.4, 117.0,  91.4,  79.3, 121.4, 123.0,  94.1,  90.4
};
x = x[1:250];
bign = nrow(x);
finish;
reset storage=procs.simprocs;
store module=normpop;
* The original deviates were generated with these lines;
n = 500;
mu=100;
sig=25;
x=normal(848393948); *set the seed;
x=normal( j(n,1) );
x = x*sig + mu;
filename tmp "norm.txt";
file tmp;
do i = 1 to n;
        put (x[i]) 6.1 "," @;
        if mod(i,10) = 0 then
                put;
end;
quit;
/* ################################################################ */
/* ----------------------------------------------------------------
        GRS.SAS
        IML routine to take a GRS sample.
   ---------------------------------------------------------------- */
%include "c:\librefs.sas";
proc iml;
start grs( S, PI_S, popvals, n, propx );
/*
   Purpose:  To draw a GRS sample of size n from the population
             popvals, with inclusion probabilities proportional
             to the vector propx.
   Inputs: popvals = NX1 vector of values from which to sample
           n = (scalar) sample size
           propx = NX1 vector of values used to compute
                   inclusion probabilities.
                   Inclusion probabilities are proportional
                   to this vector.
   Outputs: S = nX1 vector of values from popvals which
                made it into the sample
```

```
            PI_S = nX1 vector of actual inclusion
                probabilities of those in the sample
*/
        call sample( s_ind, PI_S, nrow(popvals), n, 0, 1, propx,) ;
        S = popvals[s_ind];
finish;
reset storage=procs.simprocs;
store module=grs;
quit;
/* ############################################################## */
/*
    SAMPLE.SAS
    Proc IML routine to take a randomized or systematic GRS sample.
*/
%include 'c:\librefs.sas';
%include gpmacros;
proc iml;
start sample(S,PI_SAMP, popsize, sampsize, sw1, sw2, pi1_pop, order);
/*
    Generate indices of units in simple systematic, simple random,
    general systematic, and general random sample.
    Input
    Parameters: popsize  - Size of the population (ie. N)
                sampsize - Size of sample from population (ie. n)
                sw1    - Switch 1
                 == 0 -> Random configuration: randomly sort
                         population labels before drawing sample.
                         (produces GRS samples)
                 == 1 (non-zero) -> Fixed configuration: if
                         (popsizeX1) vector order is specified,
                         order the labels according to order, if
                         order is not specified, labels are not
                         sorted and thier order is 1,2,...,N
                         (presumably, actual population values
                         would be ordered outside this routine
                         in this case)
                sw2    - Switch 2
                 = 0 -> Equal probability sample (ignore pi1)
                 = 1 (non-zero) -> Variable probability sample
                     (use pi1 vector, see below)
                pi1_pop- (optional) Vector proportional to first
                     order inclusion probabilities.
                       NOTE:
                        If sw2 =1, then this vector MUST be size
                        (popsize X 1) and is used to compute the
```

```
                        inclusion probabilities.  That is, selection
                        is proportional to this vector.
                        If sw2 =0, then the size and values of this
                        vector (if specified) is ignored.
                order - (optional) Vector (popsizeX1) by which to
                 order the population labels in systematic samples.
                 The population is ordered by horizontally appending
                 ORDER to the labels 1:N, then sorting rows on the
                 ORDER column.  See sw1 above for more.
    Output
    Parameters: s      - the sampsizeX1 vector of labels selected in
                        the sample. Values in s are integers between
                        1 and popsize.  If k is the (popsize)X1
                        population vector, and you issue run
                        sample(nrow(k), ...,s, ) then
                        the actual values in the sample are k[s].
                pi_samp -the actual 1st order inclusion probabilities
                         for the labels selected in the sample. These
                         are needed to compute Horwitz-Thompson
 estimates.
    Examples:
            Sample Type                Call
         ------------------  ------------------------
  SRS - Simple Random Sample run sample(s, pi, popN, sampn, 0, 0,,);
  GRS - Gen. Random Sample
        w/ pi prop to X      run sample(s,pi, popN, sampn, 0, 1, X,);
  SSS - Simple Systematic
        in variable X        run sample(s,pi, popN, sampn, 1, 0,, X);
  GSS - Gen. Systematic
        in X w/ pi prop to P run sample(s,pi,popN,sampn,1,1,P,X);
    Trent McDonald,
       Original GAUSS routine - 3/29/93
       Improved SAS routine   - 8/23/95
*/
/*  Check error conditions */
    if sampsize > popsize then
      do;
        file log;
        put "SAMPLE ERROR: Sample size cannot be greater than
             population size";
        return;
      end;
    else if (sw2 = 1) & (nrow(pi1_pop) ^= popsize) then
      do;
        file log;
```

```
        put "SAMPLE ERROR: Pi vector size incompatible with
             population size";
        return;
      end;
/*  Initialize */
    popu = (1:popsize)';
/*  If systematic (sw1=1) and vector order is given, sort labels
    1:N by the order vector. This gives a way to sample across
    some set of "X" values.*/
    if %present(order) & (sw1=1) then
      do;
        if (nrow(order) ^= popsize) then
          do;
            reset log;
            print "SAMPLE ERROR: Order vector not compatible
                   with population, not same size";
            reset nolog;
            return;
          end;
        popu = popu || order;
        popu = sortmat(popu,2)[,1];
      end;
/*  Use switches to set conditions */
    if sw2 = 0 then
        /* Equal probability case */
        pi1 = shape(sampsize/popsize, popsize, 1);
        /* vector of equal constants */
        /* sumc(pi) = n */
    else
      do;
        /* UnEqual probability case - Rescale the pi's to
           sum to sampsize with no pi greater than 1 */
        pi1 = scalepi( sampsize, pi1_pop );
      end;
    if sw1 = 0 then
      do;
        /* Randomly sort population indices */
        /* NOTE: random number seed should be set outside
           this routine with
           a statement like c=uniform(123456) */
        m = uniform( j(popsize,1) );
        popu = popu || pi1 || m;
        popu = sortmat(popu,3);
        pi1 = popu[,2];
        popu = popu[,1];
```

```
      end;
/*  Construct pointer vector */
    m = uniform( 1 );
    pv = m + (0:(sampsize-1))';
/*  Compute cumulative sum of pi vector */
    savep= pi1;
    pi1 = cusum(pi1);
/*  Find all indexes in pi (i) such that
    pi[i]<pv[j]<=pi[i+1] for some j */
    pi1 = shape(pi1,sampsize,popsize)';
    pv = shape(pv,popsize,sampsize);
    pop_ind = (pv > pi1)[+,] + 1;
/*  Extract unit indices from population */
    s = popu[pop_ind];
    pi_samp = savep[pop_ind];
finish;
reset storage=procs.gprocs;
store module=sample;
quit;
/* ############################################################## */
/*
        SETCOR.SAS
        IML module to produce an ordering in one vector such that
        the Spearman's rank correlation between it and some other
        vector is some specified value.
*/
%include "c:\librefs.sas";
proc iml;
reset storage=procs.gprocs;
load module=(sortmat rankcor vecvar scalepi scalex2);
start setcor( permv, finalr, x, y, r, fuzz );
/*
        Purpose: to find an ordering of the vector x such that
                the Spearman's rank corrolation between x and y is
                between r-fuzz and r+fuzz.
        Input:  x = nx1 vector
                y = nx1 vector
                r = desired Spearman's rank corr, between -1 and 1
                fuzz = how close to r you want to get
        Output: x = the re-ordered x vector
                finalr = the actual Spearman's rank corr of the
                        re-ordered x and y.
                permv = the permutation of x which produced the
                        finalr
*/
```

```
n = nrow(y);
if r <= (-1+fuzz) then
    do;
        * Sort in reverse order;
        ind = sortmat( y || (1:n)', 1)[,2];
        permv = sortmat( x || (1:n)', 1)[(n:1),2];
        permv = sortmat( ind||permv, 1)[,2];
        x = x[permv,];
        finalr = rankcor(x,y);
        put "Corr Extreme:"  finalr;
    end;
else if r >= (1-fuzz) then
    do;
        * Sort in same order;
        ind = sortmat( y || (1:n)', 1)[,2];
        permv = sortmat( x || (1:n)', 1)[,2];
        permv = sortmat( ind||permv, 1)[,2];
        x = x[permv,];
        finalr = rankcor(x,y);
        put "Corr Extreme:"  finalr;
    end;
else
    do;
        ind = sortmat( y || (1:n)', 1);
        yt = ind[,1];
        ind = ind[,2];
      * ind eventually restores original order of y;
        xt = sortmat(x,1);
        robs = rankcor( xt,yt );
        bestperm = (1:n)';
        ipermv = bestperm;
        permv = ipermv;
        iters = 300;
        i = 1;
        b = 1;
        cent = floor( n/2 );
        rt = abs(r);
        file log;
        do while(( abs(rt-robs) > fuzz ) & (i <= iters));
*               Reverse order of middle block of x;
                permv = permv[1:(cent-b)] //
                        permv[(cent+b):(cent-b+1)] //
                        permv[(cent+b+1):n];
                xtt = xt[permv,];
                robs2 = rankcor( xtt, yt );
```

```
                        if abs(rt-robs) > abs(rt-robs2) then
                           do;
                                bestperm = permv;
                                robs = robs2;
                           end;
                        put i +1 robs +1 robs2;
                        i = i+1;
                        if b>=(cent-1) then
                            do;
                                b = 1;
                                put "--------------------";
                            end;
                        else
                                b = b+1;
                end;
*               Reverse best permutation if r is negative;
                if r<0 then
                   do;
                        bestperm = bestperm[(n:1)',];
                        robs = rankcor( xt[bestperm,], yt );
                   end;
                permv = sortmat( ind||bestperm, 1)[,2];
                x = xt[permv,];
                finalr = robs;
           end;
finish;
reset storage=procs.gprocs;
store module=setcor;
quit;
/* ############################################################## */
/*
        RANKCOR.SAS
        IML module to compute Spearman's rank correlation
*/
%include "c:\librefs.sas";
proc iml;
start rankcor( x, y );
/*
  Purpose: To compute the (Spearman) rank correlation between
      two vectors of equal length.  The equation for no ties is:
                r = 1- (6* sum( d**2 ))/(n**3 - n)
      where d is the difference in ranks.
      A correction for ties is made, see Zar, p 320.

  Input:  x = nx1 vector
```

```
          y = nx1 vector

  Output: r = rank correlation, corrected for ties
*/
        n = nrow(x);
        if nrow(y) ^= n then
         return( "RANKCORR Error - Vectors must be equal length" );
        rx = ranktie(x);
        ry = ranktie(y);
        d = rx - ry;
        d = d`*d;
*       Correction for ties found in Zar;
        v = unique(x);
        tx = 0;
        do i = 1 to ncol(v);
                ti = sum( x = v[i] );
                tx = tx + ((ti**3 - ti)/12);
        end;
        v = unique(y);
        ty = 0;
        do i = 1 to ncol(v);
                ti = sum( y = v[i] );
                ty = ty + ((ti**3 - ti)/12);
        end;
        r = (n**3 - n)/6;
        r = ( r - d - tx - ty ) / sqrt( (r-2*tx)*(r-2*ty) );
        return(r);
finish;
reset storage=procs.gprocs;
store module=rankcor;
quit;
/* ################################################################ */
/*
    SORTMAT.SAS
    Proc IML routine to sort a matrix on one column.
*/
%include 'c:\librefs.sas';
proc iml;
start sortmat( x, col );
/*
    Sort a matrix on a single column.
    Inputs: x = a NXK matrix
            col = a scalar between 1 and k specifing the
                  column of x to sort on.
    Return: matrix x with rows sorted on the col-th column.
```

```
*/
    xtemp = (1:nrow(x))` || x[,col];
    create _xsort from xtemp[colname={ind key}];
    append from xtemp;
    close _xsort;
    sort _xsort by key;
    use _xsort;
    read all var {ind} into sortind;
    close _xsort;
    call delete(_xsort);
    return( x[sortind,] );
finish;
reset storage=procs.gprocs;
store module=sortmat;
quit;
/* ############################################################## */
/* ----------------------------------------------
    PROP1E.SAS
    Code to create a IML module which computes a proposed
    statistic and other neccesary quantities (like
    variance) for testing.
   ---------------------------------------------- */
%include "c:\librefs.sas";
proc iml;
start prop1e(t1, t2, rs, s1, p1, s2, p2, bign, bigm, shift, extend);
/*
    Purpose: To compute test statistics and variance estimates
             given the samples s1 and s2.
             One statistic is t=sum( z/pi )/sum(
             1/pi ) where z=I(u<v)+.5I(u=v) and pi is the correct
             inclusion probabilities for z.  Other statistic it t=
             sum(z/pi)/(N*M) where N and M are population sizes.
             Shift is a vector of shifts for the alternative
             hypothesis. Add shift to sample 2 and recompute
             the statistics.
             Extend is a vector of extention factors.
             transform sample as k(y-ymin)+ymin and
             recompute stats
    Input:   s1 = nX1 vector of values in sample 1
             p1 = nX1 vector of 1st order inclusion probabilites
                  for sample 1
             s2 = mX1 vector of values in sample 2
             p2 = mX1 vector of 1st order inclusion probabilites
                  for sample 2
             shift = kX1 vector of shifts
```

```
                extend = qX1 vector of extention factors
    Outputs: t1 = 1X(k+q) vector of statistics using known
                  bign and bigm in denom
             t2 = 1x(k+q) vector of statistics using sum of
                  weights in denom
             rs = 1x(k+q) vector of usual rank sum statistics
*/
    k = nrow(shift)*ncol(shift);
    q = nrow(extend)*ncol(extend);
    t1 = j(1,k+q);
    t2 = j(1,k+q);
    rs = j(1,k+q);
/*  Calculate 1st order inclusion probabilities of z. IE pi_z[ij]=
    pi_s1[i]*pi_s2[j] since samples are independent. */
    pz = p1*p2';
    n = nrow(s1);
    m = nrow(s2);
    s1_r = repeat(s1, 1, nrow(s2));
    do i = 1 to k;
        /* Reformat samples so can construct all possible pairs */
        s2_r = repeat(s2 + shift[i], 1, nrow(s1));
        /* Construct z=I(s1[i]<s2[j]) + .5*I(s1[i]=s2[j]) for
           all possible (i,j) */
        z = (s1_r < s2_r') + 0.5*(s1_r = s2_r');
        /* Compute statistics. Use sum(1/pz) as denom so t in
           [0,1]. Another
           choice for denom might be NM, product of population
           sizes. */
        t1[1,i] = sum( z/pz )/(bign*bigm);
        t2[1,i] = sum( z/pz )/sum( 1/pz );
        rs[1,i] = n*(2*m + n + 1)/2 - sum( z );
    end;
    ymin = s2[><,];
    do i = (k+1) to (k+q);
        s2_r = repeat(extend[i-k]*(s2 - ymin) + ymin, 1, nrow(s1));
        z = (s1_r < s2_r') + 0.5*(s1_r = s2_r');
        t1[1,i] = sum( z/pz )/(bign*bigm);
        t2[1,i] = sum( z/pz )/sum( 1/pz );
        rs[1,i] = n*(2*m + n + 1)/2 - sum( z );
    end;
/*  Note that when two indep SRS are taken,
    rs = n*(2*m + n + 1)/2 - ((n*m)/(bign*bigm)**2) * t1
*/
finish;
reset storage=procs.simprocs;
```

```
store module=prop1e;
quit;
/* ############################################################ */
/* ------------------------------------------------
    EZVAR.SAS
    Code to create a IML module to compute
    variance of my using 'EZ' memory requirements.
   ------------------------------------------------ */
%include "c:\librefs.sas";
proc iml;
start ezvar( maxap, avght, avgap, pu, pv );
/*
Purpose: to compute the average approximate variance
     for the ratio estimator.  Average is over all
     permutations of assignments of pi to units in the universes.
     ie. compute approx variance for all possible assignments
     of pi to units in both universes (N!*M! permutations)
     and average.  Actually, I have
     bypassed the permutations with the average design
     weight bit.
 Inputs: pu = (bignX1 matrix) the 1st order inclusion probabilities
     for the 1st universe.
     pv = (bigmX1 matrix) the 1st order inclusion probabilities
     for the 2nd universe.
 Output: maxap = THIS PARAMETER DOES NOTHING
     avgap = average approximate variance
     avght = average HT variance
*/
* Sort both pi vectors ;
pi1t =  sortmat( pu, 1);
pi2t =  sortmat( pv, 1);
n = nrow(pu);
m = nrow(pv);
* Average expansion factor;
wavgu = (1/pi1t)[:];
wavgv = (1/pi2t)[:];
* Average off diagonal element of puu/pu*pu matrix. Use do loops
  to save memory, may take time for large population;
wavguu = 0;
do i = 1 to (n-1);
   do j = (i+1) to n;
      wavguu = wavguu + 2*( pii1[i,j]/ (pi1t[i]*pi1t[j]));
   end;
end;
wavguu = wavguu / (n*(n-1));
```

```
wavgvv = 0;
do i = 1 to (m-1);
   do j = (i+1) to m;
      wavgvv = wavgvv + 2*( pii2[i,j]/ (pi2t[i]*pi2t[j]));
   end;
end;
wavgvv = wavgvv / (m*(m-1));
* Now compute average variance, which is (at least approximately) a
  function of average expansion factors and average 2nd order
  expansion factors.
  HERE, N AND M MUST BE EQUAL! ;
avght = wavgu*wavgv*(n*((2*n)-1)/4) + (wavgu*wavgvv + wavgv*wavguu)*
        (n*(n-1)*((2*n) - 1)/6) + wavguu*wavgvv*((3*(n**4) -
        (n*(2*n - 1)*(4*n - 1)))/12) - (n**4)/4;
* I think following two lines can work even if n not equal m;
* c is sum( (pii1/(pi1t*pi1t')) @ (pii2/(pi2t*pi2t')) );
c = 0.25 * ((wavgu*n + wavguu*n*(n-1))*(wavgv*m + wavgvv*m*(m-1)));
avgap = avght + (((n*m)**2)/4) - c;
finish;
reset storage=procs.simprocs;
store module=(ezvar);
quit;
/* ############################################################## */
/*  ------------------------------------------------
    SCALEPI.SAS
    Code to produce IML module which scales a "X" vector
    into a "Pi" vector for sampling.
    ------------------------------------------------ */
%include 'c:\librefs.sas';
proc iml;
start scalepi( n, x );
/*
    Purpose: to scale the column vector x so that it sums to
        n, yet has no elements over 1.  If p=scalepi( n, x),
        then p is the vector of (1st order) inclusion
        probabilities in a
        "pi-px" sampling design with sample size n.
    Input:  n = scalar = sample size (or sum of the result)
            x = NX1 vector to which sampling is proportional to (N
                is usually the population size)
    Return: a NX1 vector which sums to n and has no elements
        greater than 1.
*/
        _pi1 = n # (x / sum(x));
        do while( any(_pi1>1) );
```

```
            _i = loc( _pi1 >= 1 )';    /* indices where pi>=1 */
            _k = loc( _pi1 < 1)';      /* indices where pi<1 */
            _pi1[_i] = j( nrow(_i),1 ); /* truncate some to 1  */
            _pi1[_k] = (n - nrow(_i)) * (_pi1[_k] / sum(_pi1[_k]));
                                        /* rescale the others  */
        end;
        return( _pi1 );
finish;
reset storage=procs.gprocs;
store module=scalepi;
quit;
/* ############################################################## */
/*  ------------------------------------------------
    SCALEX2.SAS
    Code to produce IML module which scales a "X" vector
    so that the resulting pi vector has a variance goes up
    or down according to some parameter.
    ------------------------------------------------ */
%include 'c:\librefs.sas';
proc iml;
start scalex2( vindex, x, n );
/*
    Purpose: to scale the column vector x so that
        the resulting pi vector has variance
        indexed by vindex.  That is,
        if vindex is less than 0, exp( abs(vindex) ) is added
        to x so that variance of pi is smaller (lower bound 0) and
        if vindex is greater than zero, exp(vindex) is added to
        all x values greater than the (N-n)-th smallest x
        values (ie. added to n largest x values) so that variance
        of pi is larger (upper bound n*(N-n)/N*(N-1) ).
        If vindex is 0, nothing is done and true probability
        proportional to x sampleing is done.
 Input:  vindex = scalar = see above explaination, small
            neg numbers yeild small pi varinace,
            large positive numbers yeild large pi variance.
         x = NX1 vector to which sampling is proportional to (N
            is usually the population size)
         n = scalar = sample size
 Return: a NX1 vector changed as above.  This vector must
        later be run through scalepi. ie.
        The sampling pi is pi=scalepi( n, x2 ),
        where x2 = scalex2( ind, x, n )
*/
if vindex < 0 then
```

```
        xt = x + exp( abs( vindex ));
else if vindex > 0 then
   do;
        xt = sortmat( x, 1);
        cutx = xt[nrow(xt)-n,];
        s = loc( x >= cutx );
        xt = x;
        xt[s] = x[s] + exp( vindex );
   end;
else
        xt = x;
return( xt );
finish;
reset storage=procs.gprocs;
store module=scalex2;
quit;
/* ############################################################ */
/*  ------------------------------------------------------------
        GROUPPI2.SAS
        IML routine to construct a "grouped" PI vector such that
        there are a certain number of equal sized groups and the
        variance of the pi's is some specified number.
    ------------------------------------------------------------ */
%include "c:\librefs.sas";
proc iml;
reset storage=procs.gprocs;
load module=(vecvar scalepi scalex2 sortmat);
start grouppi2( n, bign, pctmax, ng );
/*
   Purpose: To contruct a PI vector suitable for passing to
       the GRS module such that a GRS sample of size n will
       be taken where
       (1) there are ng distinct pi's (constant pi's
       within groups), each group has an equal number
       of elements in it (ie. bign/ng)
       (2) the varinace of the pi's is pctmax*(maximum
       pi variance);
  Inputs:  n = (scalar) sample size
       bign = (scalar) population size
       pctmax = (scalar) desired percent of maximum variance for
          the pi's
       ng = (scalar) number of groups
  Return: a bignX1 vector of inclusion probabilities.
    NOTE: BIGN/NG MUST BE AN INTEGER OR THIS WON'T WORK.
*/
```

```
        fuzz = 0.00001;   * allowable percent difference ;
        maxv = n*(bign - n) / (bign*(bign-1));
        print "maxv:" maxv "target:" (pctmax*maxv);
        targ = (pctmax*maxv);
        if targ <= (fuzz/2) then  /* Call it zero > SRS sample */
           do;
                return( scalepi( n, j(bign,1) ));
           end;
        g = bign / ng; * G MUST BE AN INTEGER. g is number in each
 group;
        x = ((1:ng)') @ j(g,1);
        v1 = 0;
        pi1 = scalepi( n, x );
        vpi1 = vecvar(pi1);
        v2 = targ - vpi1;
        pi2 = scalex2( v2, x, n );
        pi2 = scalepi( n, pi2 );
        vpi2 = vecvar( pi2 );
        vpi3 = vpi1;
        file log;
        put "1 Target= " targ +1 "V(pi)= " vpi1 +1 "v1= " v1;
        i = 1;
        put (abs(vpi1 - targ)/targ );
        do while( ((abs(vpi3 - targ)/targ ) >= fuzz) & (i<=50));
                v3 = (targ - vpi1)/(vpi2 - vpi1) * (v2-v1) + v1;
                pi3 = scalex2( v3, x, n );
                pi3 = scalepi( n, pi3);
                vpi3 = vecvar( pi3 );
                if vpi3 < targ then
                   do;
                        vpi1 = vpi3;
                        v1 = v3;
                   end;
                else
                   do;
                        vpi2 = vpi3;
                        v2 = v3;
                   end;
                i = i+ 1;
                put i +1 "Target= " targ +1 "V(pi)= " vpi3 +1
    "v= " v3;
        end;
        return( pi3 );
finish;
reset storage=procs.simprocs;
```

```
store module=grouppi2;
quit;
/* ################################################################ */
/* ------------------------------------------------
    OVERPIJ.SAS
    Code to create a IML module which computes Overton's
    second order inclusion probabilities.
   ------------------------------------------------ */
%include "c:\librefs.sas";
proc iml;
start overpij( p, sw );
/*
    Compute Overton's approximate second order inclusion
    probabilities.
    This routine computed Overton's approximation to
    second order inclusion probabilities.  Overton proposed them
    in OSU Tech Report 114 (1985) and Stehman and Overton
    investigated them in JASA(1994), 89, p30-43.
 Input: p = nx1 vector of first order inclusion probabilites
    sw = scalar switch, If upcase(sw) is anything other than
         "POP", p is assumed to be the sample probabilities
         (ie. n = nrow(p)).  If upcase(sw) = "POP", then p is
         assumed to be a correctly scaled population pi vector
         (ie. n = sum(p)).
 Return: a nXn symetric matrix, with p (1st order inclusion
    probabilities) on the diagonal and Overton's approximate 2nd
    orders as the off-diagonal elements.
*/
    if upcase(sw) = "POP" then
        n = sum(p);
    else
        n = nrow(p);
    nr = nrow(p);
    w = 1 / p;
    /* Overton's 2nd order approx. Pi_ij = 1/w_ij */
    p2 = (2*(n-1)) / (2*n#(w*w`) - shape(w,nr,nr)
         - shape(w,nr,nr)`) ;
    p2 = diagrv( p2, p );
    return( p2 );
finish;
reset storage=procs.gprocs;
store module=overpij;
quit;
/* ################################################################ */
/* ------------------------------------------
```

```
        SIMMACS.SAS
        Some macros used in simulation.
   ------------------------------------------- */
%macro storit( npsave );
/*
  Purpose: To store iteration results.  I must use a macro,
    inserted into MAIN module so that symbol table is correct
    and GETVALS (and VALUE()) work right.
  Input global macro variables:
    &itervar = the variable holding current iteration number
    &isavlist = list of global variables to save each iteration
    &iresmat = name of iteration result matrix
    &numpsave = number of iterations between writing to disk
    &nameroot = name of catalog into which all results go
  Output: none
*/
        if &itervar = 1 then
                &iresmat =  &isavlist ;
        else if &itervar <= &numiter then
                &iresmat = &iresmat //  &isavlist ;
        if mod( &itervar, &npsave ) = 0 then
           do;
                &icolnm=&isavnm;
                reset storage=results.&nameroot ;
                mattrib &iresmat colname=&icolnm
                  label="Iteration Statistics";
                store &iresmat;
                store &icolnm;
                reset storage=tmp;
           end;
%mend;
/* ---------------------------------------------------------- */
%macro storsi;
/*
        Purpose: To store overall simulation results. Must use a
           macro so that referencing is correct.
        Input macro vars:
            &ssavlist = list of global variables to save
            &sresmat = name of iteration result matrix
            &nameroot = name of catalog into which all results go
        Output: none
*/
        &sresmat = &ssavlist ;
        &scolnm = &ssavnm;
        /* Open a catalog and save */
```

```
        reset storage=results.&nameroot;
        mattrib &sresmat colname=&scolnm rowname=&srownm
           label="Overall Simulation Results";
        store &sresmat;
        store &scolnm &srownm;
%mend;
/* ------------------------------------------------------------ */
%macro parstor;
/*
  Purpose: to store all macro variables in the catalog
  Input: none really, this uses a list of all macro variables
     by calling %stormvar with each one.  I can't figure a
     better way to do this.
  Output: none. values of macro variables are stored in character
     matrices of same name as macro variable
*/
reset storage=results.&nameroot;
%stormvar( simuroot );
%stormvar( popln1   );
%stormvar( popln2   );
%stormvar( n        );
%stormvar( m        );
%stormvar( design1  );
%stormvar( design2  );
%stormvar( estimate );
%stormvar( alpha    );
%stormvar( itersum  );
%stormvar( iresmat  );
%stormvar( sresmat  );
%stormvar( srownm   );
%stormvar( simusum  );
%stormvar( itervar  );
%stormvar( dispit   );
%stormvar( seed     );
%stormvar( setseed  );
%stormvar( numiter  );
%stormvar( nameroot );
%stormvar( numpsave );
%stormvar( isavnm   );
%stormvar( isavlist );
%stormvar( icolnm   );
%stormvar( ssavnm   );
%stormvar( ssavlist );
%stormvar( scolnm   );
%stormvar( shft     );
```

```
%stormvar( ext      );
%stormvar( PCTVAR1  );
%stormvar( PCTVAR2  );
%stormvar( cor1     );
%stormvar( cor2     );
%stormvar( NGROUP   );
store pop1;
store pivec1;
store pop2;
store pivec2;
store avghtv;
store avgapv;
store truer1 truer2;
%mend;
/* ------------------------------------------------------------ */
%macro stormvar( x );
/*
        Purpose: to make a IML matrix named &x and store the
                 value of
                 the macro variable named &x in the matrix and
                 store the matrix in the results catalog.
        Assume the correct catalog is already open.
*/
        &x = "&&&x";
        store &x;
%mend;
/* ############################################################## */
/* ------------------------------------------------
    DISPITER.SAS
    Code to create a IML module which displays iterations
    results on the screen
   ------------------------------------------------ */
%include "c:\librefs.sas";
proc iml;
start dispiter( res, labs );
/*
    Purpose: to display results of an iteration somehow.
    Input: res  = a 1Xc vector containing the values to display
           labs = a 1Xc vector of labels for the values in res
    Output: none
*/
        /* Define a window */
        window iterate cmndline=cmnd
           rows=6
           msgline="Current iteration of McDonald's simulation"
```

```
        group=line1
                #1
        group=rtrn
              /
        group=itlabel
              +2 labs p=yes
        group=itvals
              +2 res p=yes;
        labs = right(labs);
        display iterate ( #1
                "Current Iteration of McDonald's simulation" / ),
                iterate.itlabel repeat,
                iterate (/),
                iterate.itvals repeat;
finish;
reset storage=procs.simprocs;
store module=dispiter;
quit;
/* ############################################################## */
/* ----------------------------------------------
    DIAGRV.SAS
    Code to create a IML module which puts a vector
    on the diagonal of a square matrix. Or, puts the
    diagonal elements of one matrix on the diagonal of
    another.
   ---------------------------------------------- */
%include "c:\librefs.sas";
proc iml;
start diagrv( sqmat, diagele );
/*
    Purpose: to put the nX1 vector DIAGELE on the diagonal of the
        nXn matrix SQMAT, or to put the diagonal elements of
        the nXn
        matrix DIAGELE on the diagonal of the nXn matrix SQMAT.
    Input: sqmat = a nXn matrix
           diagele = a nX1 vector or a nXn matrix
    Return: a nXn matrix with diagele (or diagonal elements
        of diagele) on the diagonal of sqmat.
    A SAS error is generated (something like non-conformable) if
    sqmat and diagele are not the right sizes.
*/
    n = nrow(sqmat);
    ans = sqmat#(j(n,n) - i(n)) + diag(diagele);
    return( ans );
finish;
```

```
reset storage=procs.gprocs;
store module=diagrv;
quit;
/* ################################################################ */
/*  ------------------------------------------------------
        SETRAND.SAS
        IML module to set the ransom number sequence.
    ------------------------------------------------------ */
%include "c:\librefs.sas";
proc iml;
start setrand( s );
/*
        Purpose: to initialize the random number sequence.
        Input: s = scalar seed
        Output: none
*/
        _k = uniform( s );
finish;
reset storage=procs.simprocs;
store module=setrand;
quit;
/* ################################################################ */
/* ------------------------------------------------
    SUMSIMU.SAS
    Code to create a IML module which sumarizes entire
    simulation results
   ------------------------------------------------ */
%include "c:\librefs.sas";
proc iml;
start sumsimu( simvals, rownam, mat );
/*
    Purpose: to sumarize a simulation.  E[t], E[Var(t)], and
        size of the test will be computed.
    Input: mat = matrix containing all the iteration results
    Output: simvals = a 2Xncol(mat) matrix containing the simulated
                      means and variances of each column of mat
            rownam = a 1x2 string matrix of row names for the simvals
*/
        /* compute means of columns */
        simvals = mat[:,];
        /* compute variances of columns */
        ussq = mat[##,];
        n = nrow(mat);
        v = (ussq - n#simvals##2)/(n-1);
        simvals = simvals // v;
```

```
        rownam = {"Mean (E[])" "Variance (n-1)"};
finish;
reset storage=procs.simprocs;
store module=sumsimu;
quit;
/* ############################################################ */
/* ------------------------------------------------
    SUMITER.SAS
    Code to create a IML module which sumarizes results from
    a single iteration.
   ------------------------------------------------ */
%include "c:\librefs.sas";
proc iml;
start sumiter( iter, vlist );
/*
    Purpose: to sumarize an iteration somehow.  This might involve
        displaying, testing, or further calculations.
Results should be stored using STORITER.
    Input: iter = the number of the current iteration
           vlist = a vector containing the names of variables
   to display
    Output: none
*/
        vvals = vlist;
        vlist = {x y z};
        window itersum cmndline=cmnd
            group=itlab
                #1 "Iteration"
            group=itnum
                #2 iter
            group=vlabel
                +2 vlist
            group=itvars
                +2 vvals;
        display itersum.itlab noinput, itersum.vlabel
                repeat noinput,
                itersum.itnum noinput, itersum.itvars
                repeat noinput;
finish;
reset storage=procs.simprocs;
store module=sumiter;
reset storage=procs.gprocs;
load module=getvars;
quit;
/* ############################################################ */
```

```
/* ------------------------------------------------
    TESTP1.SAS
    Code to create a IML module which implements the
    testing algorithm for the PROP1 statistic.
   ------------------------------------------------ */
%include "c:\librefs.sas";
proc iml;
start testp1( t  );
/*
    Purpose: At present, nothing.  This is simply a dummy
    procedure.  Could test here. I choose to test outside
    simulations in TESTS.SAS.
*/
     k = t;
finish;
reset storage=procs.simprocs;
store module=testp1;
quit;
/* ############################################################ */
/* ------------------------------------------------
    VEC.SAS
    Code to create a IML module which takes a matrix
    and makes it into a vector.
   ------------------------------------------------ */
%include "c:\librefs.sas";
proc iml;
start vec( x );
/*
    Purpose: to make a matrix into a vector.
    Input: x = nXm matrix
    Return: a vector of size nmX1 containing the elements of x in
            in ROW-MAJOR order.
*/
    return( shape( x, nrow(x)*ncol(x), 1 ));
finish;
reset storage=procs.gprocs;
store module=vec;
quit;
```

The simulations (code above) simply computed test statistics and stored them. The following routine actually constructed the tests and computed p-values.

```
/*  ----------------------------------------------------------
```

```
        TESTS.SAS
        Compute tests from a simulation, sizes, power, etc.
        Produce some data files for graphing.
    ------------------------------------------------------------ */
options mprint symbolgen;
%let droot=c:\trent\ranksamp\sassimu;
%include "&droot\librefs.sas";
* Name of the data set test results are to be stored in;
%let dsname=save.ctour5n;
libname glib "c:\graphing";
libname save "c:\graphing";
proc iml;
* ------------------------------------------------------------ ;
start testt2(rej, t2, muhat, vhat, alpha );
/*
        Test the statistic with 1/piu*piv in denominator.
        (t_p in the paper)
        Compare to a Beta distribution.
        Let iters = number of iterations in simulation
        Input: t2 = (itersX1) vector of the simulated statistics
               muhat = true mean of statistic under null =
                       .5 usually
               vhat = (approx) var of statistic computed in
                      simulation using EZVAR = ratiovar
               alpha = nominal level of test = p-value cutoff
        Output: rej = size of the test = percent of rejections
*/
*       Estimate Beta params by MOM ;
        a= muhat*( (muhat*(1-muhat)/vhat) - 1 );
        b= muhat*( ((1-muhat)**2/vhat) + 1 - (1/muhat));
        print a b;
*       Compute p value;
        if (a <= 1.25) then
           do;
                print "a<1.25: Normal used";
                p = probnorm( (t2 - muhat)/sqrt(vhat) );
           end;
        else
                p = probbeta( t2, a, b );
*       Make a decisions;
        decision = (p<(alpha/2)) | ((1-p)<(alpha/2));
        rej = decision[:];
finish;
* ------------------------------------------------------------ ;
start testw( r_fpc, r_inf, rs, n, m, bn, alpha );
```

```
/*
    Purpose: to do the Wilcoxon tests.  One test is done with
        usual variance, the other is done with the ties and
        finite population correction in the variance.  Both
        compute the test assuming
        the null hypothesis of equal CDF's and equal population
        sizes.
    Input: rs= iters X 1 vector of wilcoxon rank sum statistic
           n = size of sample 1
           m = size of sample 2
           bn = size of both populations (N)
           alpha = the nominal size of the test
    Output: r_fpc = size or percent of rejections using the
                    FPC corrected var
            r_inf = size or percent of rejections using the
                    usual Wilcoxon var
*/
        /* Compute the two variances */
        vfpc = varsrs( n, m, bn );
        vinf = (n*m*(n+m+1))/12;
        rsmean = n*(n+m+1)/2;
        /* do the tests */
        zfpc = (rs - rsmean)/sqrt(vfpc);
        zinf = (rs - rsmean)/sqrt(vinf);
        pfpc = probnorm( zfpc );
        pinf = probnorm( zinf );
        r_fpc = (pfpc<(alpha/2)) | ((1-pfpc)<(alpha/2));
        r_inf = (pinf<(alpha/2)) | ((1-pinf)<(alpha/2));
        r_fpc = r_fpc[:];
        r_inf = r_inf[:];
finish;
* ------------------------------------------------------------- ;
start mkdset( outmat, vnames);
*
        Make a permanent SAS data set from some vectors for
        later plotting by another routine, like GCONTOUR.
        Inputs: outmat = matrix to save in data set
                vnames = matrix of strings to use for
                         variable names
;
        create &dsname from outmat[colname=vnames];
        append from outmat;
        close &dsname.;
        use &dsname.;
        show datasets;
```

```
        show contents;
        list all;
finish;
* ------------------------------------------------------------ ;
* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%;
*  Macro to get p-value for each shift or extension;
* %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%;
%macro testtyp1( simu );
        simufile = "&simu";
        reset storage=simufile;
        load;
        print avgapv avghtv;
*       Put the shifts in a numeric matrix, this is a
        little tricky;
        s = shft;
        call change( s, "{"," ");
        call change( s, "}"," ");
        s = s + " A";
        p = indexc( s, "0123456789");
        shifts = -1;
        do while( p > 0);
                s = substr( s, p );
                e = indexc( s, " ");
                shifts = shifts // num(substr( s, 1, e-1));
                s = substr( s, e+1);
                p = indexc( s, "0123456789" );
        end;
        shifts = shifts[2:nrow(shifts)];
*       Put the extensions in a numeric matrix, this is
        a little tricky;
        s = ext;
        call change( s, "{"," ");
        call change( s, "}"," ");
        s = s + " A";
        p = indexc( s, "0123456789");
        exts = -1;
        do while( p > 0);
                s = substr( s, p );
                e = indexc( s, " ");
                exts = exts // num(substr( s, 1, e-1));
                s = substr( s, e+1);
                p = indexc( s, "0123456789" );
        end;
        exts = exts[2:nrow(exts)];
        print shifts exts;
```

```
*       extract shift from extension columns;
        nshifts = nrow(shifts) ; * number of shifts and
             extensions calculated in file;
        nexts = nrow(exts);
        altres  = iterres[,(1:nshifts) ||
               ((nshifts+nexts+1):(2*nshifts+nexts)) ||
               ((2*nshifts+2*nexts+1):(3*nshifts+2*nexts)) ];
*       COMPUTE SIZES OF THE TESTS;
        alph = 0.05;
        file log;
        srt2 = -1*j(nshifts,1);
        srwfpc = -1*j(nshifts,1);
        srwinf = srwfpc;
        do j = 1 to ncol(altres);
                statind = ceil( j/nshifts );
                sftcol = mod(j, nshifts );
                if sftcol = 0 then sftcol = nshifts;
                stats = altres[,j];
                put "Statistic:" statind 3. " Shift #:"
                   sftcol 3. " Max Statistic: " (stats[<>,]) @;
                if statind = 2 then
                 do;
                  call testt2( rejt2, stats, 0.5, avgapv, alph );
                  put  " Size: " rejt2 @;
                  srt2[ sftcol ] = rejt2;
                 end;
                else if statind = 3 then
                 do;
                  call testw( rejfpc, rejinf, stats, num(n),
                       num(m), nrow(pop1), alph );
                  put " Size w/fpc: " rejfpc " Size w/o fpc:"
                      rejinf @;
                  srwfpc[ sftcol ] = rejfpc;
                  srwinf[ sftcol ] = rejinf;
                 end;
               put;
        end;
        put;
        put "=========================================";
        put;
*       Now do extensions;
        altres  = iterres[,((nshifts+1):(nshifts+nexts))||
              ((2*nshifts+nexts+1):(2*nshifts+2*nexts)) ||
             ((3*nshifts+2*nexts+1):(3*nshifts+3*nexts)) ];
        ert2 = -1*j(nexts,1);
```

```
        erwfpc = -1*j(nexts,1);
        erwinf = srwfpc;
        do j = 1 to ncol(altres);
                statind = ceil( j/nexts );
                sftcol = mod(j, nexts );
                if sftcol = 0 then sftcol = nexts;
                stats = altres[,j];
                put "Statistic:" statind 3. " Extend #:"
                 sftcol 3. " Max Statistic: " (stats[<>,]) @;
                if statind = 2 then
                 do;
                  call testt2( rejt2, stats, 0.5, avgapv, alph );
                  put  " Size: " rejt2 @;
                  ert2[ sftcol ] = rejt2;
                 end;
                else if statind = 3 then
                 do;
                  call testw( rejfpc, rejinf, stats, num(n),
                  num(m), nrow(pop1), alph );
                  put " Size w/fpc: " rejfpc " Size w/o fpc:"
                  rejinf @;
                  erwfpc[ sftcol ] = rejfpc;
                  erwinf[ sftcol ] = rejinf;
                 end;
               put;
        end;
%mend;
* ------------------------------------------------------
        reset storage=procs.gprocs;
        load module=(vecvar rankcor);
        reset storage=procs.simprocs;
        load module=varsrs;

*  EXAMPLE CALL;
        %testtyp1( results.examp1 );
        call mkdset( shifts || srt2 || srwfpc || srwinf ||
                     exts || ert2 || erwfpc || erwinf,
                     {"shift" "s_t" "s_fpc" "s_nfpc" "ext"
                     "e_t" "e_fpc" "e_nfpc"} );
quit;
```